VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

NÁVRH ROZHRANÍ LASEROVÉ ŘEZAČKY S POUŽITÍM ROZŠÍŘENÉ REALITY
LASER CUTTER INTERFACE WITH AUGMENTED REALITY ELEMENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Matej Kajan**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Zemčík**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Matej Kajan ***ID:*** 211151

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Návrh rozhraní laserové řezačky s použitím rozšířené reality

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout kamerový systém pro lokalizaci a segmentaci materiálu umístěného v laserové řezačce. Navrhnout rozhraní k umístění vypalovaného vzoru na materiál s použitím prvků počítačového vidění a rozšířené reality.

1. Proveďte rešerši rozšířené reality v souvislosti s uživatelskými rozhraními.
2. Seznamte se se segmentačními metodami v počítačovém vidění.
3. Navrhněte vlastní hardwarové řešení kamerového systému.
4. Navrhněte postup zpracování obrazových dat.
5. Navrhněte uživatelské rozhraní s prvky rozšířené reality.
6. Implementujte HW a SW nástroje.
7. Připravte technologický demonstrátor.

DOPORUČENÁ LITERATURA:

[1] - HLAVÁČ, Václav a Milan ŠONKA, 1992. Počítačové vidění. Praha: Grada. ISBN

80-854-2467-3.

[2] - JÄHNE, Bernd a Horst HAUSSECKER, 2000. Computer vision and applications:

a guide for students and practitioners. San Diego: Academic Press. ISBN 01-

237-9777-2.

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Tomáš Zemčík

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

Abstrakt

Rozhranie laserovej rezačky s použitím rozšírenej reality dovoľuje rýchlejšiu a bezpečnejšiu realizáciu rezania. Tento návrh je uskutočnený pomocou metód počítačového videnia spojených s návrhom grafického užívateľského rozhrania.

Kľúčové slová

Rozšírená realita, laserová rezačka, počítačové videnie, používateľské rozhranie, segmentácia, kalibrácia kamery, spracovanie obrazu, trekovanie, perspektívna transformácia, afínna transformácia, SVG, XML, Qt, C++, ArucoMarkery, G-Code, OpenCV, GRBL

Abstract

User interface of a laser cutter with elements of augmented reality, allows for a faster and safer execution of the process of cutting. This proposal is accomplished by using methods of computer vision together with the design of a graphical user interface

Keywords

Augmented reality, laser cutter, computer vision, user interface, segmentation, camera calibration, image processing, tracking, perspective transform, affine transform, SVG, XML, Qt, C++, ArucoMarkers, G-Code, OpenCV, GRBL

Bibliografická citácia:

KAJAN, Matej. Návrh rozhraní laserové řezačky s použitím rozšířené reality [online]. Brno, 2021 [cit. 2021-05-19]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/134714. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Ing. Tomáš Zemčík.

Prehlásenie autora o pôvodnosti diela

Meno a priezvisko študenta:	*Matej Kajan*
VUT ID študenta:	*211151*
Typ práce:	*Bakalárska práca*
Akademický rok:	*2020/2021*
Téma záverečnej práce:	*Návrh rozhrania laserovej rezačky s použitím rozšírenej reality*

Prehlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúceho záverečnej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďaľej prehlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si naplno vedomí následkov porušenia ustanovení § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovení časti druhej, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

.

V Brne dňa: 21. mája 2021 ----------

podpis autora

Poďakovanie

Chcel by som poďakovať vedúcemu bakalárskej práce Ing. Tomášovi Zemčíkovi za odbornú pomoc, nápady, cenné rady a najmä za nesmiernu ústretovosť pri vývoji tejto práce.

V Brne dňa: 21. mája 2021 ----------

podpis autora

OBSAH

Zoznam symbolov a skratiek

Skratky:

FEKT	…	Fakulta elektrotechniky a komunikačních technologií
VUT	…	Vysoké učení technické v Brně
VAL	...	Visually Augmented Laser cutting (Vizuálne rozšírené laserové rezanie)
AR	…	Augmented reality (Rozšírená realita)
VR	…	Virtual reality (virtuálna realita)
OLI	…	outside-looking-in
ILI	…	inside-looking-in
SLAM	…	Simultaneous Localization and Map Building
GUI	…	Graphical user interface (grafické užívateľské rozhranie
FOV	…	Field of view (uhol záberu)
SVG	...	Scalable vector graphics

Zoznam obrázkov

Zoznam tabuliek

1 ÚVOD

Laserové rezačky sú čoraz viac relevantné v rôznych oblastiach a stali sa dôležitým nástrojom na budovanie modelov a prototypov. VAL (Visually Augmented Laser cutting) navrhuje novodobý prístup k použitiu rozšírenej reality ako vizuálnu spätnú väzbu priamo na pracovnú plochu[1]. Rozšírená realita nám umožňuje kombinovať interaktívnu počítačovú grafiku s reálnym prostredím, ako je napríklad pracovná plocha. [2] Tento prístup znižuje časovú náročnosť, redukuje chyby a vytvára podporu pre nové postupy. Taktiež okamžitá vizuálna spätná väzba pomáha užívateľovi lepšie vyhodnotiť svoj dizajn a následne ho upravovať. [3] Rovnako použitím kamery sa zvyšuje bezpečnosť pri práci, keďže užívateľ nemusí byť fyzicky prítomný v okolí lasera. Ďaľej, vďaka rozhraniu bude umožnené narábať s laserovou rezačkou aj neskúseným osobám, laikom a zväčší sa tak spektrum použitia.

Účelom práce je teda navrhnúť grafické používateľské rozhranie s prvkami rozšírenej reality na interakciu s laserovou rezačkou. V jednotlivých kapitolách definujem základnú terminológiu a oblasti, ktoré budú využívané na jeho realizáciu. V druhej kapitole sa zaoberám rozšírenou realitou ako takou, čitateľovi sa snažím utvoriť obraz o rôznych technológiách a prístupoch v rámci rozšírenej reality. Oblasť počítačového videnia, metódy segmentácie ako aj potrebná kalibrácia kamery je diskutovaná v tretej kapitole. V štvrtej kapitole definujem prostriedky na realizácia, od hardvérových, cez firmvér, až po softvér použitý pri vývoji. Kapitola päť opisuje spôsob implementácie prostriedkov ako aj dve riešenia ako by bolo možné pristupovať k naplneniu zadania. Šiesta kapitola diskutuje vylepšenia, ktoré by bolo možné implementovať. Záver hodnotí dosiahnuté výsledky a popisuje formu konečného riešenia.

2 ROZŠÍRENÁ REALITA

Rozšírená realita (AR – Augmented Reality) je technológia, ktorá dovoľuje počítaču generovať virtuálny obraz tak, aby presne prekryl fyzické objekty v realite. Na rozdiel od virtuálnej reality (VR), kde používateľ je kompletne ponorený do virtuálneho sveta a kde s reálnym svetom nemanipuluje, v AR je schopný pôsobiť na virtuálnu časť pomocou reálnych objektov. [4]

Uvádzajú sa tri nutné predpoklady pre AR :

1. Kombinácia reálneho a virtuálneho sveta
2. Možnosť interakcie v reálnom čase
3. 3D rekonštrukcia

Tieto predpoklady definujú technické požiadavky na realizáciu AR systému. Displej pre zobrazovanie reality s virtuálnymi objektami. Systém, ktorý generuje interaktívnu grafiku pre používateľa, ktorá odpovedá na jeho požiadavky v reálnom čase.

Trekovací systém, ktorý určuje pózu užívateľa vzhľadom na referenčný bod a aktualizovanie tejto pózy v čase. [4]

2.1 AR trekovacie technológie

Tretia požiadavka je spojená so schopnosťou systému zakotviť virtuálny obsah do reálneho sveta tak, aby bol vnímaný ako reálny objekt.

Aby bolo možné registrovať virtuálny obsah v realite, musí sa definovať vzťah (pozícia a orientácia) používateľa k objektu záujmu. V závislosti od aplikácie a technológii , kotviaci objekt môže byť napríklad fyzický objekt (značka...) alebo definovaná lokácia. [4]

Teda registrácia v 3D rovine sa skladá z dvoch fáz:

Registračná fáza, ktorá určí polohu používateľa a objektu.
Stopovacia, *Sledovacia fáza*, ktorá aktualizuje vzájomnú polohu používateľa a kotviaceho objektu. [4]

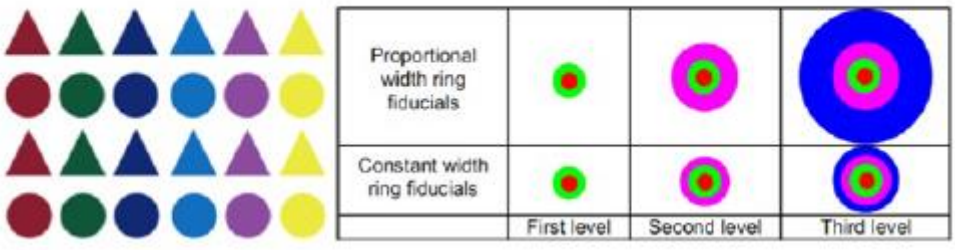
2.1.1 Trekovanie založené na videní

Definuje sa ako prístup, ktorý určí zachytávanie dát pomocou kamerového senzoru. Tento spôsob je čoraz viac populárny vďaka minimálnym požiadavkám na hardware, vďaka zvyšovanej výpočetnej sile a všadeprítomnosti mobilných zariadení a tabletov, ktoré sú už vybavené kamerou a displejom (čo z nich robí ideálnych kandidátov). [4]

Infračervené trekovanie - Niektoré z najstarších techník zahrňovali použitie „cieľov“, od ktorých sa svetlo odrážalo, čo zjednodušilo detekciu vďaka ich vysokému jasu v porovnaní s prostredím. Tieto „ciele“ mohli byť namontované na stopovaný objekt, ktorý bol priamo snímaný externou kamerou, definované ako „outside-looking-in“ (OLI), alebo niekde v prostredí a samotná kamera bola na danom objekte, tiež „inside-looking-in“ (ILI). ILI konfigurácia ponúkala vyššiu presnosť a vyššie rozlíšenie oproti OLI. Aj keď infračervené trekovanie umožňuje zväčšovanie ohľadne pracovného priestoru a precíznosť, či robustnosť k iluminačným efektom, je tento spôsob komplexný, drahý a zložitý z dôvodu fyzickej infraštruktúry. [4]

Trekovanie s pomocou viditeľného svetla - Senzory založené na viditeľnom svetle sú najčastejšie optické senzory v kombinácii s kamerami, ktoré nájdeme v laptopoch, smartfónoch apod.. Pre AR systémy sú teda nanajvýš použiteľné, keďže môžu byť použité, jednak pre prostredie reálneho sveta ako aj pre registrovanie virtuálneho obsahu, ktorý sa doň premieta. [4]

Východiskové trekovanie - Východiskové z dôvodu pridania umelých orientačných bodov (fiducial markers) do prostredia, aby pomohli pri zachytávaní a stopovaní. Účelom je, aby vznikol referenčný bod alebo bod merania. Prvotný princíp používal malé LED-ky alebo farebné značky, ktoré boli zachytené porovnávaním farieb. Poloha a orientácia kamery mohla byť určená z počtu známych pozícií týchto orientačných bodov. Benefit tohto spôsobu bola možnosť pridania ďalších bodov počas vyhodnocovania a pomocou známych polôh predošlých bodov určiť ich pozíciu. Týmto spôsob sa dalo dynamicky rozširovať pracovný priestor. [4]



Obrázok 2-1 Orientačné body **[4]**

Trekovanie podľa prirodzenej vlastnosti - Východiskové trekovanie vyžaduje modifikáciu prostredia, aby boli realizovateľné, čo z praktického hľadiska nemusí byť žiadané alebo vôbec možné. So zvyšujúcou sa výpočetnou silou nastala možnosť zachytiť polohu kamery s použitím vlastností prirodzeného (reálneho) prostredia. Algoritmami počítačového videnia sa detekujú vlastnosti v zachytených snímkach, ktoré sú unikátne v blízkom okolí, ako napríklad body, rohy a prienik čiar. Pre každú z týchto vlastností existuje „descriptor", ktorý identifikuje a rozlíši jednotlivé vlastnosti okolia a pomocou algoritmov podobným východiskovému trekovaniu (orientačné body) sa kalkuluje poloha kamery. [4]

Trekovanie založené na modeloch - Jedny z prvých pokusov používali primitívne 3D modely vytvorené ručne, pomocou aproximácií ako kombinácia jednoduchých prvkov, priamky, kruhy, valce atď.. Extrahovaním dát zo snímanej scény a porovnaním s daným 3D modelom sa vyhodnocovala pozícia kamery. Neskôr kombináciou s rozpoznávaním prirodzených vlastností objektov a pridaním textúr sa zväčšila robustnosť v komplexných a premenlivých prostrediach. V poslednej dobe sa uvažuje o technikách, ktoré súčasne vytvoria a aktualizujú mapu reálneho prostredia a zároveň sa samy lokalizujú. Napríklad takzvaný SLAM (Simultaneous Localization and Map Building), ktorý nevyžaduje predvytvorený 3D model. Bol najskôr navrhnutý pre navigáciu robotov v neznámom prostredí, ale bol neskôr adaptovaný pre AR. [4]

Ďalšie prístupy používajú technológie, ktoré získavajú 3D informácie o bodoch v priestore, napr. KinectFusion používa dáta z hĺbkového optického senzoru, ktorý vytvára kvalitný 3D model reálneho sveta a tie sú následne použité na trekovanie pozície Kinect-u v priestore.

Iné techniky -

Magnetické trekovanie zariadenia používajú vlastnosti magnetických polí na kalkuláciu vzájomnej polohy transmitter – receiver (napr. objekt – používateľ).

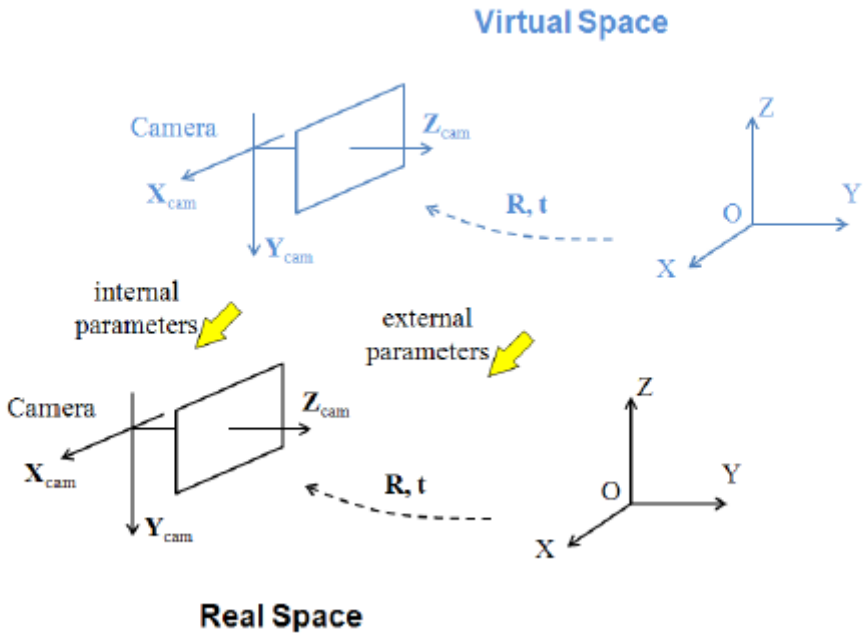
Inerciálne trekovanie používa akcelerometre, gyroskopy a magnetometre na určenie realtívnej polohe a orientácii trekovaného objektu.

GPS trekovanie je možné využiť v externých prostrediach, problém nastáva s presnosťou GPS (odchýlka v jednotkách metra) a rušením signálu okolitým prostredím.

Hybridné trekovanie spočíva v spájaní dát z viacerých a rôznych druhov senzorov pre zvýšenie presnosti a prekonanie slabých stránok konkrétnych trekovacích metód. [4]

2.2 Kombinovanie reálnych a virtuálnych obrazov

K docieleniu konečnej AR vizualizácie je nutné zabezpečiť kalibráciu kamery, registrovanie, trekovanie a kompozíciu prostredia. [4] Kalibrácia zahrňuje zhodu v parametroch tak, aby počítačovo vykreslený obraz skutočne zodpovedal reálnemu zobrazeniu scény. Poznáme interné parametre, ktoré určujú ako 3D snímaný obraz sa premietne do 2D obrazu, zatiaľ čo externé parametre definujú pozíciu a orientáciu kamery v priestore.



Obrázok 2-2 Kalibrácia kamery **[4]**

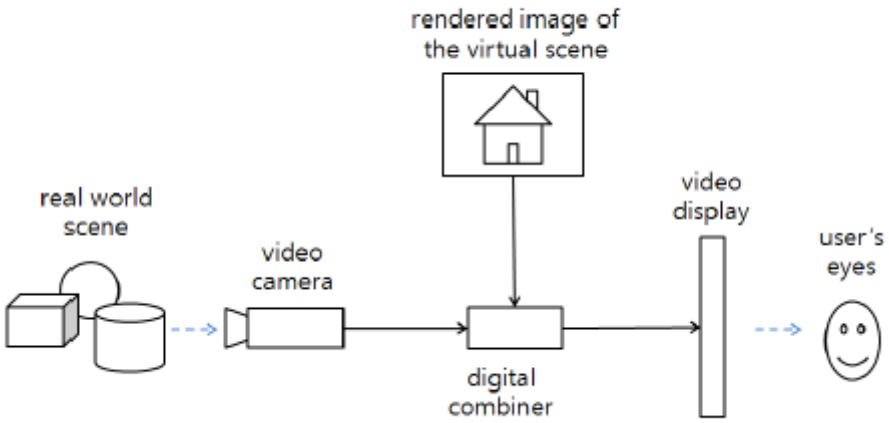
Pre interné parametre platí proces porovnávania geometrických vlastností 3D a výsledného 2D obrazca. Externé parametre sa vymedzia pomocou trekovania. Pokiaľ sa jedná o statickú scénu, je potrebné určiť pozíciu kamery relatívne k referenčnému rámcu. Zvyčajne sú techniky založené na počítačovom videní.

Trekovanie sa zaslúži o pozíciu a orientáciu cieľových objektov v rámci súradnicovej sústavy danej trekovacím systémom. Aby virtuálna scéna bola správne zarovnaná s reálnym prostredím, musí sa súradnicový systém pre virtuálny obraz zhodovať so súradnicovým systémom vo fyzickom prostredí. Tento proces sa nazýva registrovanie. [4]

Po splnení predošlých požiadaviek nasleduje kompozícia, teda syntéza a zobrazenie virtuálneho a reálneho obsahu užívateľovi.

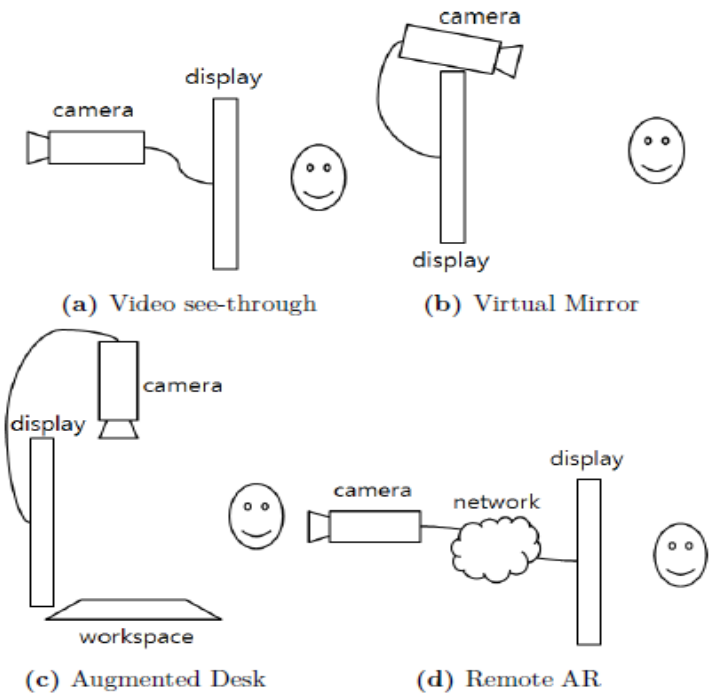
2.2.1 Video zobrazovanie

Tento spôsob zahrňuje digitálny proces na kombinovanie virtuálnej a reálnej zložky scény. Najprv systém digitalizuje reálny svet kamerou, aby následne kombinovaný obraz mohol byť vykreslený. [4]



Obrázok 2-3 Štruktúra zobrazenia pomocou kamery **[4]**

Zvyčajne býva kamera vložená v blízkosti samotného displeja, čo má za následok ilúziu, že používateľ sleduje svet cez konkrétny displej (a). Ďalšie spôsoby môžu vytvárať virtuálne zrkadlo (b), sledovanie na diaľku (d) alebo pohľad zhora na pracovnú plochu (c). [4]

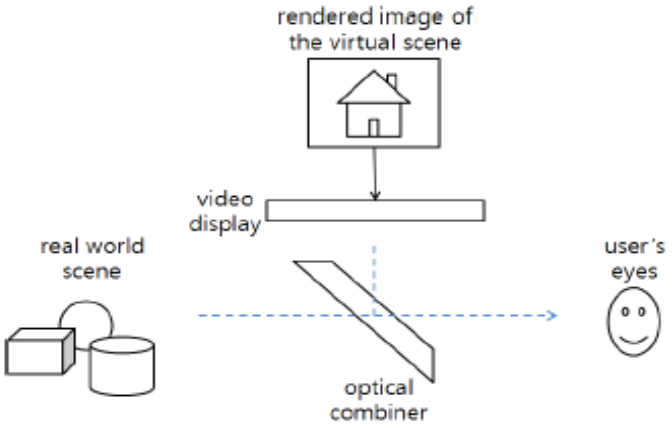


Obrázok 2-4 Variácie konfigurácií displej-kamera-objekt/používateľ **[4]**

Tento spôsob zobrazovania je široko používaný najmä z nenáročných hardverových požiadaviek a jednoduchej implementácii, či už na PC alebo mobilnej platforme. Jedna z výhod je schopnosť presne riadiť proces kombinovania reálnej a virtuálnej scény, a algoritmy počítačového videnia alebo zobrazovanie vo viacerých vrstvách dokážu vložiť virtuálny obsah na reálne obrazce. Za nevýhodu môžeme považovať nepriamy pohľad na reálny svet (cez kamera-displej), s čím je spojené obmedzenie rozlíšenia , skreslenie alebo oneskorenie obrazu [4].

2.2.2 Opticky priehľadné zobrazovanie

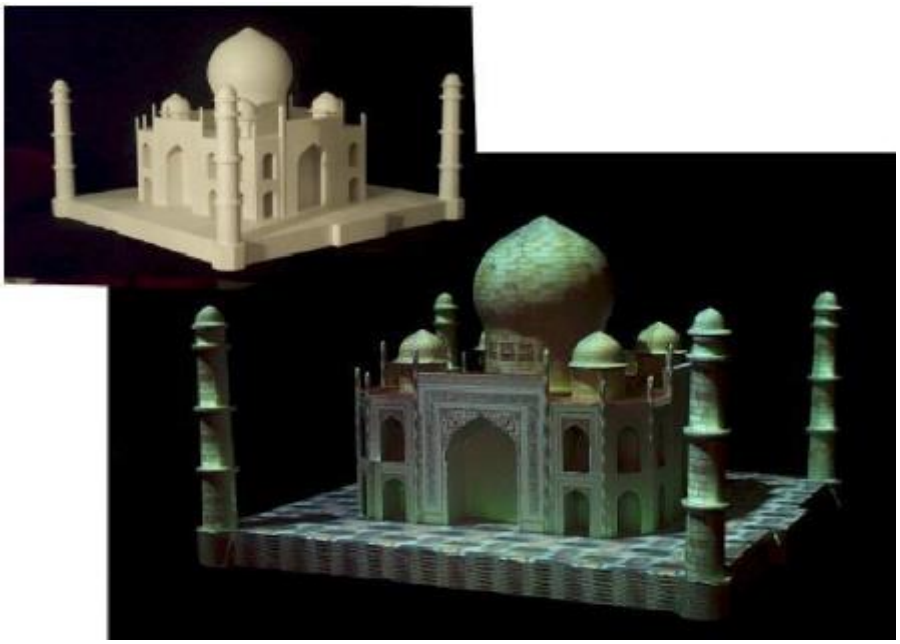
V tomto prípade sa zlučuje virtuálny obsah s reálnym pohľadom. Typickým zástupcom je HUD (head up display), ktorý sa používa v kokpitoch alebo v moderných autách. Iné formy používajú transparentné LCD displeje, ako napríklad HMD (head-mounted display). Výhoda tohto zobrazovania je sprostredkovanie priameho pohľadu na reálny svet, čiže nemusíme uvažovať limitáciu rozlíšenia alebo skreslenia. Naopak, môže vznikať nesprávne usporiadanie („missalignement“) medzi reálnym a virtuálnym svetom alebo časové oneskorenie [4].



Obrázok 2-5 Opticky priehľadné zobrazovanie **[4]**

2.2.3 Projekcie

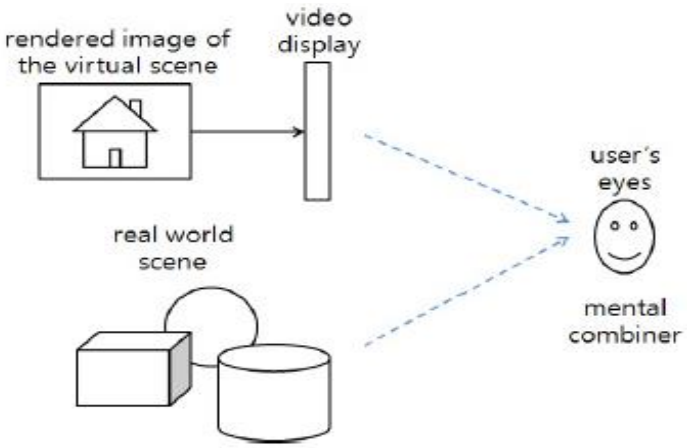
Zatiaľ čo predošlé metódy kombinovali reálny a virtuálny obrazec, projekcia premieta virtuálny obsah priamo na fyzický objekt. Nie je teda potrebné zahrnutie používateľa v rámci použitia senzorov, naopak, kombinácia so sledovaním pohľadu používateľa a objektu, na ktorý sa premieta, môže vznikať interaktívna AR. Najväčším obmedzením je prirodzene nutnosť použiť projektor, ktorý je fixovaný na jednu konkrétnu pozíciu, čo teda znemožňuje dynamicky meniť prostredie. Projekcie sú taktiež náchylné na svetelné podmienky [4].



Obrázok 2-6 Projekcia Taj Mahal-u **[4]**

2.2.4 Multiplexované zobrazovanie

Na rozdiel od metód, ktoré priamo kombinovali reálny a virtuálny obsah, v tomto prípade je celý proces ponechaný na užívateľa. Virtuálna scéna je zaznamenaná do fyzického prostredia, avšak vykreslený obraz nie je zložený s realitou. Keďže sa nezlučujú obidva obsahy, nekladie sa vysoký nárok výpočetnú silu a je zhovievavý k nepresnej registrácii medzi oboma zložkami [4].



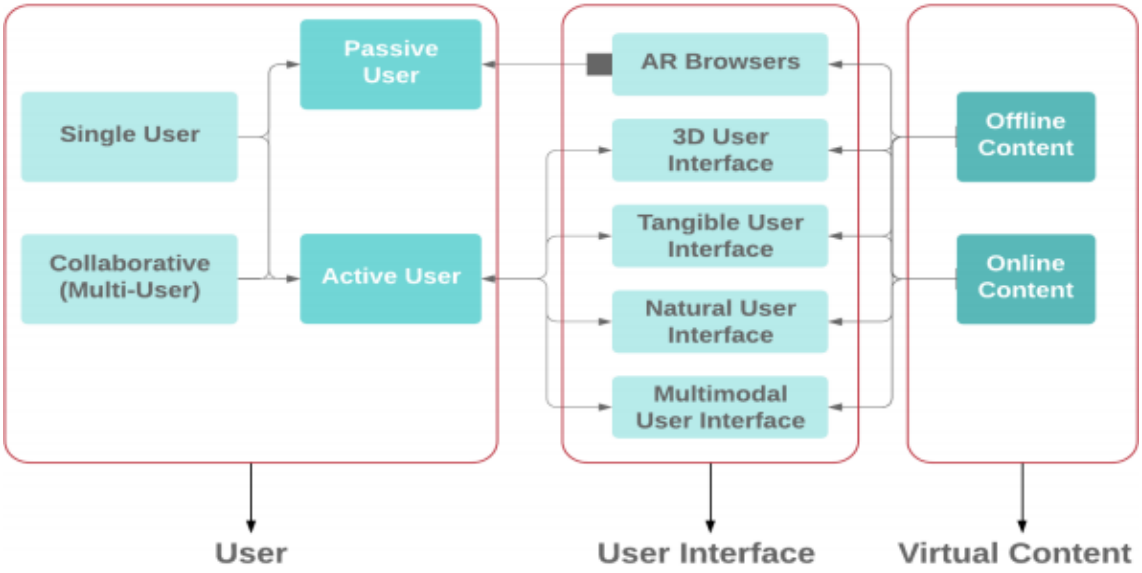
Obrázok 2-7 Multiplexované zobrazovanie **[4]**

2.3 AR technológie na interakciu a manipuláciu

V závislosti od aplikácie alebo konkrétnej implementácie AR technológie berieme do úvahy tri základné časti AR v súvislosti s interakciou :

1. používateľ
2. používateľské rozhranie
3. virtuálny obsah [5]

Dané používateľské rozhranie definuje spôsob prístupu k obsahu, ako aj schopnosť a prostriedky človeka pôsobiť na virtuálny obsah. Použite AR z pohľadu človeka môže nastať jednotlivo alebo kolaboračne, a z pohľadu prístupu aktívne, kde človek pôsobí na virtuálny obsah cez dané rozhranie, a pasívne, kde používateľ hrá úlohu pozorovateľa a iba vníma virtuálny obsah [5].



Obrázok 2-8 AR aspekty interakcie **[5]**

Z pohľadu prístupu používateľa :

- *pasívny používateľ* dostáva podnety (audio-vizuálne, haptické), ale ďaľej nemôže manipulovať s virtuálnym obsahom.

- *aktívny používateľ* môže kedykoľvek zasiahnuť a upravovať alebo manipulovať s virtuálnym obsahom. Technika interakcie ďaľej definuje spôsob, akým používateľ je schopný ovplyvniť obsah, napríklad gestom alebo reálnym fyzickým objektom [5].

Z pohľadu použitia osôb:

- *Jednotlivo,* čiže existuje iba jeden používateľ, ktorý je schopný vidieť a poprípade manipulovať virtuálny obsah.

- *Kolaboratívne* znamená prítomnosť viacerých používateľov naraz. Prostredie je teda zdieľané. Z tohto hľadiska je potrebné zabezpečiť schopnosť interakcie v reálnom čase a vysokorýchlostné pripojenie. V niektorých prípadoch je taktiež nutné určiť prioritu možnosti interakcie pre jednotlivých používateľov [5].

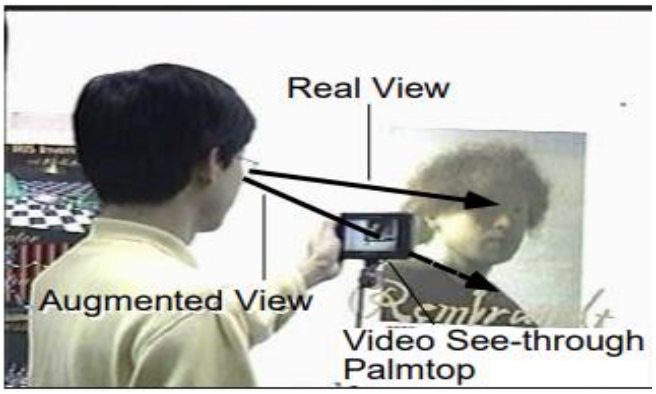
2.3.1 AR rozhrania

V tejto sekcii diskutujem metódy pristupovania a interakcie s AR. Sú definované rôzny typy rozhraní od tradičných 2D, myš, klávesnica, dotykový displej, alebo 3D a iné metódy, ako napríklad gestá, reč. Rôzne typy rozhraní sú, prirodzene, použité na základe konkrétnej aplikácie a implementácie.

AR informačný prehliadač - Je to pojem, ktorý reprezentuje spôsob, kde AR displej je považovaný za okno do informačného priestoru. V zásade používateľ pohybovaním v priestore alebo manipuláciou displeja skúma AR prostredie. Zvyčajne na interakciu sa používajú tradičné nástroje ako klávesnica, myš alebo dotykový displej. Je to najrozšírenejší spôsob rozhrania, keďže z pohľadu používateľa je jednoduchý na použitie. Naopak, spôsob interakcie s AR obsahom je limitovaný.

V porovnaní s VR, kde navigačné rozhranie je potrebné manévrovať v 3D virtuálnom priestore, AR systémy zobrazujú virtuálne objekty do reálneho sveta. Používatelia teda nepotrebujú ďalšie rozhranie na manévrovanie, ale používajú prirodzené zručnosti na pohyb v prostredí. Iné formy interakcie zahŕňajú výber informácií na prezeranie, filtrovanie informácií, detailný pohľad alebo zmena štýlu vizualizácie. Zvyčajne sa používajú tradičné 2D grafické užívateľské rozhrania (GUI) a obrazovka, teda na interakciu sa využívajú tradičné nástroje ako klávesnica, myš, joystick alebo dotykový displej [4].

Tento typ rozhrania je jeden z najdôležitejších a jeden z najpoužívanejších. Metóda interakcie je jednoduchá na naučenie a používatelia sú schopní pracovať s bežnými rozhraniami (mobil, počítač...). Príkladom by mohol byť jeden z prvých AR systémov, NaviCam (1995), pozostával z LCD displeja a CCD-kamery, ktorá zachytávala reálnu zložku a tá sa následne spojila s počítačovo generovaným obrazcom [4].



Obrázok 2-9 NaviCam AR prehliadač **[6]**

3D rozhranie - Umožňuje užívateľovi manipulovať s virtuálnym obsahom cez priamy 3D input vo fyzickom alebo virtuálnom svete. Boli zhrnuté typy 3D interakcií do troch kategórií – navigovanie (manévrovanie), výber a manipulácia. Zatiaľ čo manévrovanie nemusí byť priamo aplikovateľné na AR systémy, výber a manipulácia môžu byť ľahko prevzaté. V mnohých prípadoch sú použité zariadenia so šiestimi stupňami voľnosti manipulácie virtuálnych objektov (translácia a rotácia). Existujú rôzne typy zariadení, ktoré boli vytvorené pre VR a 3D používateľské rozhrania, napríklad, 3D myš alebo paličkové typy ukazovadiel. Z nich najpoužívanejšie sú 3D senzory sledovania pohybu. Dovoľujú trekovať rôzne fyzické objekty, vrátane pohybov používateľa, tým pádom mu dovoľuje manipulovať a ukazovať na virtuálne objekty [4].

Zariadenia založené na hmate sú ďalším typom pre 3D rozhrania. Nielenže fungujú ako ukazovadlá, ale taktiež poskytujú spätnú väzbu vo forme sily a hmatu, čo ešte viac komplementuje vizuálny zážitok vytváraním ilúzie fyzickej podoby virtuálneho objektu [4].

3D rozhrania teda zabezpečujú kvalitnú interakciu v prirodzenej a známej podobe. Istá analógia medzi použitím vo VR a AR existuje, avšak metódy pre manipuláciu s virtuálnymi objektami sa líšia od tých s fyzickými objektami. Vo VR používateľ musí držať zariadenie na riadenie manipulácie s virtuálnym obsahom, zatiaľ čo v AR sa priamo manipuluje s fyzickými objektami a používateľ využíva svoje ruky na manipuláciu [4].

Hmatateľné rozhrania - Používajú fyzické objekty ako reprezentácia virtuálnych entít a ako premostenie medzi fyzickým a virtuálnym svetom. Na manipulovanie virtuálneho obsahu sa teda používajú reálne objekty v priestore. Napríklad pomocou paličky môžeme vyberať, pohybovať, otáčať a vo všeobecnosti manipulovať s virtuálnym obsahom. Podstatou je teda použitie rozhrania ako input na interakciu a následne AR na vizualizáciu obsahu, kde obidva obsahy plynulo splývajú. Definujú sa dve charakteristiky pre toto rozhranie – každý virtuálny objekt je registrovaný na objekt fyzický a interakcia spočíva v manipulácii fyzického objektu [7]. To sprístupňuje použitie trekovania objektov, pomocou algoritmov počítačového videnia. Potom nielenže systém dokáže identifikovať a rozpoznať daný objekt, ale aj pozorovať jeho pohyb v priestore, čo sa využije na spomínané interakcie [5].

Prístupy, akým je rozhranie navrhnuté sú dva (alebo ich kombinácia). Prvým je priestorovo multiplexované rozhranie, kde každý fyzický nástroj má len jednu funkciu, a časovo multiplexované rozhranie, kde daný nástroj je použitý pre rôzne funkcie v závislosti na okolnostiach a kontexte. Prvý spôsob je viac intuitívny pre používateľa, zatiaľčo časovo multiplexované rozhranie je viac dynamické, tým pádom ale zložitejšie na implementáciu [4].

Jedným z problémov je ukázať používateľovi, akým spôsobom sa transformujú pohyby fyzického objektu na povely, respektíve ich interpretácia. Prirodzene to vedie ku kombinovaniu používateľovho inputu s gestami a hlasom, čo vedie potom k multimodálnym rozhraniam [8].

Obrázok 2-10 Hmatateľné AR rozhranie **[4]**

Prirodzené rozhranie - Je založené na rozpoznávaní pohybov a gést. Vlastne prostriedkom na interakciu sa stáva samotný človek, respektíve časti tela. S pokročením technológií počítačového videnia AR systémy začali byť schopné rozpoznávať pohyby tela a gestá v reálnom čase bez potreby nejakých senzorov na tele človeka. Navyše s komerčnou dostupnosťou kamier, táto forma interakcie sa stala viac a viac používanou pri VR alebo AR [4].

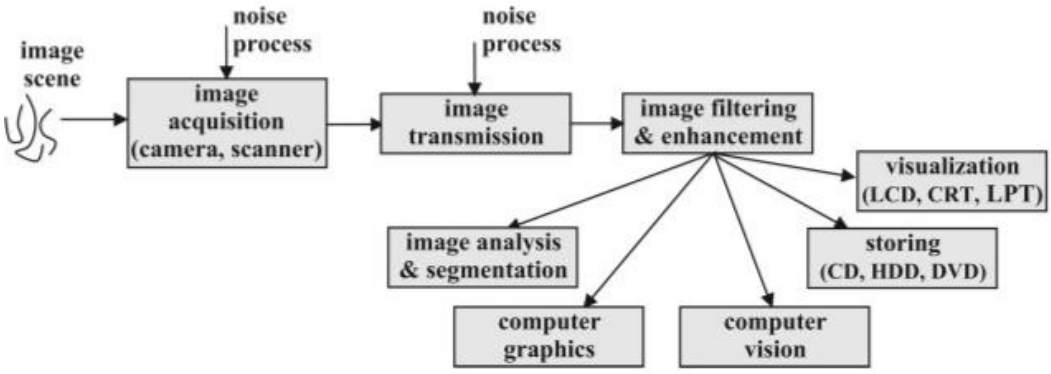
Napriek všetkým technologickým pokrokom zostáva priestor na zlepšenie. Úlohu, ktorú treba vyriešiť, je aj naďalej limitovaná presnosť a robustnosť tejto metódy. Ďalším problémom je vlastne porovnanie efektivity tohto spôsobu v porovnaní s ostatnými jednoduchšími metódami [4].

Obrázok 2-11 Trekovanie prstov pre rozhranie na základe projekcie **[4]**

Multimodálne metódy - Spočívajú v kombinácii rozličných metód. Prirodzenou kombináciou je napríklad reč spolu s gestami. Táto dvojica sa navzájom doplňuje, kde reč je použitá na kvantitatívne príkazy a gestá na kvalitatívne. Týmto spôsobom je napríklad možné prekonať problém s presnosťou pri interakcii pomocou pohybov a gést tým, že ho doplníme o rečové príkazy. Avšak použitím viacerých metód interakcie môžeme ochromiť používateľa alebo zhoršiť jeho zážitok. Okrem iného sa tým zvyšuje komplexnosť implementácie, ako aj pridanie ďalšej vrstvy zložitosti pri samotnom vývoji [9].

3 POČÍTAČOVÉ VIDENIE

Počítačové videnie (CV – Computer Vision) sa snaží umelo nahradiť ľudský vizuálny aparát, a tým dovoliť počítačom pochopiť a interpretovať vizuálne dáta zo statických snímkov alebo videí. CV zahŕňa úlohy, ako napríklad *akvizícia digitálneho obrazu,* kde senzor zosníma a prenesie informácie na vytvorenie obrazu. *Spracovanie alebo digitalizácia obrazu,* kde sa vytvorí digitálna reprezentácia obrazu a *analýza obrazu,* kde sa vyberú podstatné informácie z obrazu alebo sa obraz modifukuje podľa požiadaviek. Pre ľudí je spracovanie a rozpoznanie vizuálnych dát jednoduchá úloha, avšak pre počítače sa obrazové dáta interpretujú ako sieť čísel. Problémom je strata informácie pri prechode z 3D snímaného obrazu do 2D interpretácie pre počítač, ktorý dokonca môže podliehať šumu alebo skresleniu. Tieto deformácie obrazu môžu byť spôsobené prírodnými podmienkami, nedokonalosťami v šošovkách, mechanickým nastavením alebo elektrickým šumom v samotnom snímači [10].



Obrázok 3-1 Reťaz spracovania obrazu **[11]**

Jeden z dôvodov prečo nemáme všeobecný systém pochopenia obrazu, je ten, že 2D obraz môže reprezentovať potencionálne nekonečné množstvo možností. Vnímanie obrazu je v podstate psychologické, čiže je ťažko spracovateľné čisto analytickou metódou. Teda každý matematický algoritmus musí byť doplnený o význam alebo kontext daného obrazu [12].

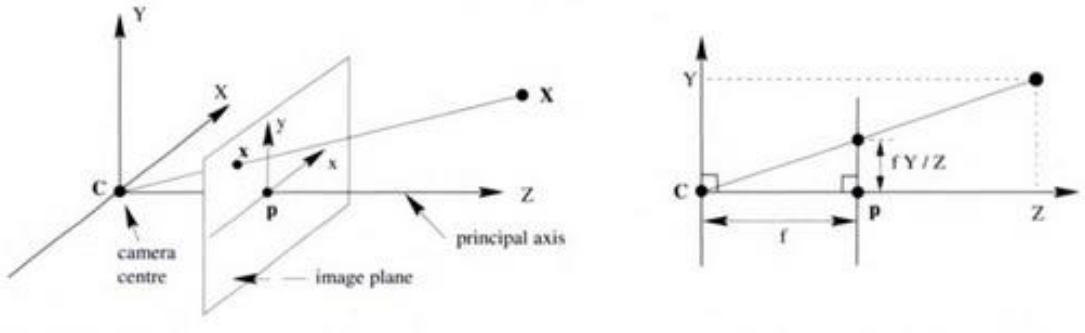
3.1 Kalibrácia kamery

Dôležitým krokom pre správne určenie vzťahu medzi obrazom objektu a jeho fyzickými parametrami je kalibrácia kamery. Odhadujú sa parametre kamery, ako napríklad parametre šošovky alebo obrazového senzoru. Tie môže použiť na korekciu skreslenia, meranie rozmeru objektov alebo určenie polohy kamery v scéne.

Ako už bolo spomenuté v kapitole 2.2, rozdeľujeme parametre kamery na interné (intrinsic) a externé (extrinsic). Najprv je však potrebné definovať, všeobecne, model kamery.

3.1.1 Model Kamery

Najjednoduchším modelom kamery je dierková kamera („pinhole camera“), ktorej model opisuje matematický vzťah medzi 3D bodom v snímanom priestore a jeho projekciou na obrazovú rovinu.



Obrázok 3-2 Geometria dierkovej kamery **[13]**

Uvažujme premietanie 3D bodov na rovinu, kde centrom premietania je počiatok súradnicového systému a rovinu Z = *f*, čo je obrazová rovina. Bod **X** (3D) = $(X, Y, Z)^T$, je mapovaný na bod obrazovej roviny pomocou priamky, teda jej prienikom cez rovinu. Bod $(X, Y, Z)^T$ je mapovaný na bod $(f\,X/Z, fY/Z, f)^T$, čo reprezentuje mapovanie medzi reálnym svetom a súradnicami obrazu [13].

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \Rightarrow \begin{pmatrix} f\,X/Z \\ fY/Z \end{pmatrix} \qquad \textbf{(3.01)}$$

Homogénne súradnice nám umožnia prejsť z 2D do 3D priestoru pridaním ďalšej súradnice (= 1), tým pádom sme schopní popísať transformácie obrazu ako maticové násobenie. Ak je teda, reálny svet a body obrazu reprezentované ako homogénne vektory, potom projekcia je lineárne mapovanie medzi danými bodmi.

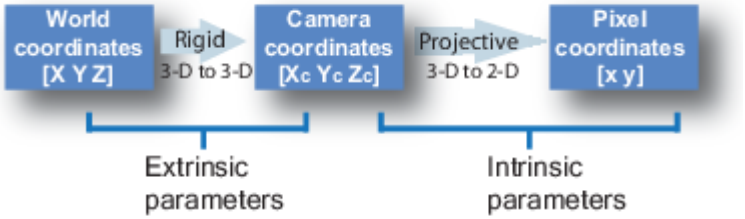
$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \textbf{(3.02)}$$

Zjednodušením zápisu vznikne :

$$x = P\mathbf{X} \tag{3.03}$$

kde **X** je reprezentácia reálneho bodu v priestore a „x“ je bod obrazu [13]. Matica „P“ (3x4) reprezentuje maticu kamery [14], tiež označovanú ako **C**.

3.1.2 Parametre kamery



Obrázok 3-3 Parametre kamery **[15]**

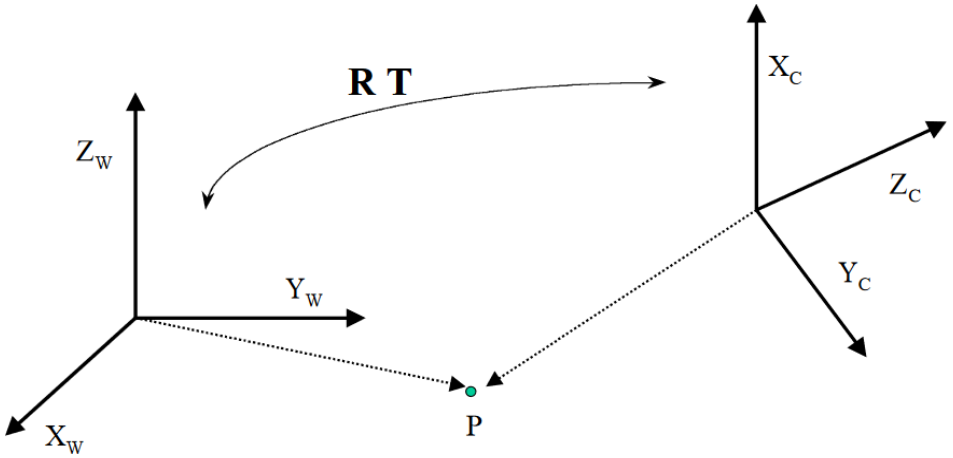
Interné parametre kamery definujú päť parametrov pre konkrétny model kamery. Tieto parametre nám umožňujú presne premietnuť 3D obraz na súradnice 2D obrazu.

$$K = \begin{pmatrix} f.m_x & \gamma & u_0 \\ 0 & f.m_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.04}$$

f..... ohnisková vzdialenosť
u_0, v_0..... reprezentujú stred obrazu
γ..... skosenie
m_x, m_y..... inverzia šírky a výšky pixelu na projekčnú plochu

Parametre u_0, v_0 sú zvyčajne blízke 0, ak principiálny bod je vlastne stredom obrazu. Taktiež parameter γ je rovnako zvyčajne rovný 0.

Externé parametre obrazu určujú polohu a orientáciu kamery v reálnom svete. Poloha kamery sa určuje translačnou maticou a jej orientácia, maticou rotácie.



Obrázok 3-4 Reprezentácia externých parametrov kamery **[16]**

Translačný pohyb reprezentuje matica **t** vo všetkých smeroch [17].

$$t = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \quad \textbf{(3.05)}$$

Orientácia kamery sa určuje pomocou vektora **R**, ktorá rotuje korešpondujúce osi každého obrazu do seba.

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad \textbf{(3.06)}$$

Pre úplnosť sa rotačná matica dá rozpísať aj pre všetky osi samostatne [16].

3.1.3 Matica kamery

Spojením všetkých parametrov vzniká konečná matica kamery, s 11 nezávislými parametrami kamery. Formálne to reprezentuje vzťah medzi 3D bodom a jeho projekciou na obrazovú rovinu [16].

$$\mathbf{P} = K\,[R \mid t] \quad \textbf{(3.07)}$$

$$\mathbf{P} = \begin{pmatrix} f.m_x & \gamma & u_0 \\ 0 & f.m_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \quad \textbf{(3.08)}$$

3.2 Segmentáciu obrazu

Hlavným cieľom segmentácie je rozdelenie obrazu na časti tak, aby sme získali súvis s objektmi alebo oblasťou reálneho sveta, ktoré obsahuje daný obraz. Čiže je založená na istých kritériách tak, aby rozdelila obraz na časti, ktoré sú pre nás zaujímavé. Čiastočná segmentácia teda rozdeľuje obraz na oblasti s rovnakou vlastnosťou ako je jas, farba alebo textúra.

Segmentačné metódy sme schopní rozdeliť do troch skupín. Prvou je *globálna znalosť* o obraze, čo je reprezentované histogramom vlastností obrazu. Druhou sú metódy založené na detekcii hraníc oblastí a tretia popisuje postupy priamo určujúce oblasti. Teda každá oblasť sa dá reprezentovať jej uzavretou hranicou [10].

3.2.1 Prahovanie obrazu

Najjednoduchší segmentačný postup. Mnoho objektov je charakterizovaných tak, že ich povrch má konštantnú (podobnú) odrazivosť alebo absorbciu svetla. Môžeme teda definovať konštantu jasu alebo prah, aby sme určili rozdiel medzi objektom a pozadím.

Platí teda, že pre vstupný obraz f transformovaný na výstupný obraz g :

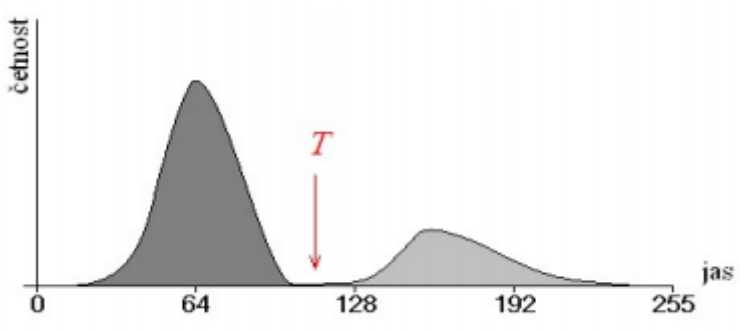
$$g(i,j) = 1 \text{ ; pre } f(i,j) >= T \qquad \textbf{(3.09)}$$
$$= 0 \text{ ; pre } f(i,j) < T$$

Kde T je stanovený prah, g(i,j) = 1 pre oblasť objektu a g(i,j) = 0, pre pozadie [10].

Táto metóda je teda vhodná v prípade, keď sa objekty nedotýkajú a ich hodnoty odtieňu sú rozlíšiteľné. Vtedy je možné určiť *globálnu* hodnotu prahu. V praxi ale nemusí nastať, že jeden objekt má jednotný odtieň šedej po celom jeho povrchu, čo môže byť , spôsobené nerovnomerným osvetlením alebo šumom. [18] V tomto prípade je nutné použiť rôzne hodnoty prahu pre segmentáciu. Tento typ prahovania sa nazýva *adaptívny,* v ktorom sa hodnota prahu mení *lokálne* na základe skúmanej oblasti objektu.

Hodnotu prahu je možné určiť rôznymi metódami. Najjednoduchšie je mať predošlú informáciu o obraze. Napríklad, akú percentuálnu plochu zaberá hľadaný objekt (-y) alebo priamo farba daného objektu.

Pomocou *histogramu* sme schopní určiť počet hodnôt jasov pre daný obraz. Hodnota hľadaného prahu je lokálne minimum medzi dvoma maximami. [10]



Obrázok 3-5 Histogram obrazu **[10]**

Táto metóda je vhodná pre bi-modálne histogramy, teda pre histogramy s dvoma maximami. Pri vyššom počte maxím nám vznikne väčšie množstvo miním, čiže získavame viac prahov. Potom pre jednotlivé hodnoty prahov je výsledok segmentácie odlišný.

Multispektrálne prahovanie sa používa, keď je informácia o obraze uchovaná v rôznych spektrách, napríklad farebný obraz. V zásade sa oblasť rovnakej farby určí ako skupina pixelov s rovnakými alebo veľmi podobnými hodnotami v daných spektrách. Prah sa určuje jednotlivo pre všetky spektrá, následne sa výsledok skombinuje do spoločného segmentovaného obrazu a vyhodnotí.

3.3 Priestorové transformácie

Priestorová transformácia definuje geometrický vzťah medzi korešpondujúcimi bodmi vstupného a výstupného obrazu. Vstupný obraz obsahuje referenčné body, ktorých poloha a hodnota je vopred známa. Výstupný obraz zahŕňa warpované dáta [21].

$$[X,Y] = [\, X(u,v), Y(u,v) \,] \qquad \textbf{(3.10)}$$

$$[u, v] = [U(x,y), V\,(x,y) \,] \qquad \textbf{(3.11)}$$

[u,v] sa odkazuje na koordináty vstupného obrazu a [x,y] výstupného obrazu.
U a V sú ľubovoľné mapovacie funkcie.

Mapovanie môže byť dané buď ako vzťah výstupného súradnicového systému k vstupnému alebo naopak.

Geometrické transformácie sa snažia o elimináciu geometrických skreslení. Táto transformácia je vektorová funkcia **T**, ktorá mapuje vstupný obraz do výstupného bod po bode v 2D. Pri aplikovaní transformácia sa najprv mapuje bod (pixel) vstupného obrazu do výstupného a následne sa interpoluje hodnota jasu, ktorá sa danému bodu priradí.

Zvyčajne predpis pre geometrickú transformáciu sa určuje pomocou polynomiálnej aproximácie [10].

$$x' = \sum_{r=0}^{n} \sum_{k=0}^{n-r} a_{rk} * x^r * x^k \tag{3.12}$$

$$y' = \sum_{r=0}^{n} \sum_{k=0}^{n-r} b_{rk} * x^r * x^k \tag{3.13}$$

Transformácia mapovaného bodu x', respektíve y', tak závisí na oboch súradniciach vo vstupnom obraze. Pokiaľ sú polohy pôvodných bodov a bodov mapovaných známe, sme schopní riešením sústavy lineárnych rovníc určiť vzťah pre transformáciu.

Afínna transformácia je najjednoduchšia a jej pomocou dokážeme aplikovať typické geometrické transformácie ako rotácia, translácia, skosenie a škálovanie. Na definovanie afínnej transformácie – koeficientov - nám v tomto prípade stačí trojica bodov.

$$\begin{aligned} x' &= a_0 + a_1 x + a_2 y \\ y' &= b_0 + b_1 x + b_2 y \end{aligned} \tag{3.14}$$

Afínnu transformáciu je možné rovnako reprezentovať pomocou 3x3 transformačnej matice.

$$[\mathrm{u, v, 1}] = [\mathrm{x, y, 1}] * \begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{pmatrix} \tag{3.15}$$

Vynásobením vektora [x, y, 1], reprezentovaného v homogénnych súradniciach, s transformačnou maticou, dostávame výsledný vektor [u, v, 1] [21].

Perspektívna transformácia, spolu s projekciou na rovinu zobrazenia tvoria perspektívnu projekciu, tiež nazývanou ako homografia, ktorá používa homogénne súradnice [22].

$$\tilde{x}' = \tilde{H}\tilde{x} \tag{3.16}$$

$\tilde{\boldsymbol{H}}$ značí ľubovoľnú 3x3 maticu, ktorá je homogénna. Výsledkom je homogénna súradnica $\tilde{\boldsymbol{x}}'$, ktorá musí byť normalizovaná, aby sme získali nehomogénny výsledok [22].

$$x' = \frac{h_{00}x+h_{01}y+h_{02}}{h_{20}x+h_{21}y+h_{22}}$$

$$y' = \frac{h_{10}x+h_{11}y+h_{12}}{h_{20}x+h_{21}y+h_{22}} \tag{3.17}$$

4 PROSTRIEDKY NA REALIZÁCIU

V tejto kapitole definujem nástroje použité pri vývoji aplikácie. V hardvérových prostriedkov diskutujem o požiadavkách na kameru, laserovej rezačke a rôznych firmvéroch. V softvérových prostriedkoch opisujem podstatné prvky na realizáciu od GUI, SVG až po OpenCV.

4.1 Hardvérové prostriedky

4.1.1 Kamerový aparát

Cieľom práce bolo vytvoriť GUI pre bežného človeka, čiže nároky na hardvér by mali byť minimálne a hardvér jednoduchý na použitie. Významným prvkom bude samotná kamera. Preto je potrebné zadefinovať potrebné parametre na určenie kategórie kamery. Keďže sa ale jedná o statickú scénu, nepredpokladajú sa vysoké nároky na parametre kamery. Z toho dôvodu bola práca vyvíjaná pomocou bežne dostupnej webkamery, ktorú by bolo možno pripojiť cez bežné rozhranie USB.

Parametre kamery podstatné pre realizáciu:

- Rozlíšenie
- Uhol záberu (FOV – field of view)
- Skreslenie
- Cena (marginálny parameter)

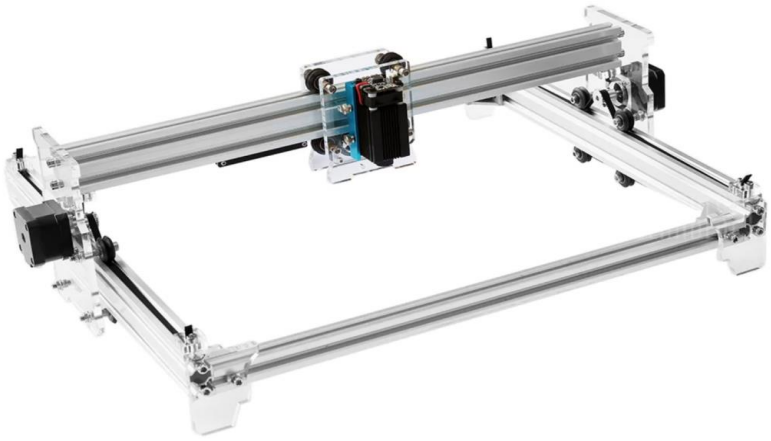
Pre minimálnu chybu umiestňovania objektov je nutné čo najvyššie rozlíšenie kamery. Keďže je potrebné definovať vzťah px~mm, vyššie rozlíšenie zabezpečí jemnejší prepočet, a tým pádom menšiu chybu pre konečné umiestnenie.

Uhol záberu definuje polohu – výšku kamery nad rezacou rovinou. Pri vysokej hodnote FOV kameru nie je potrebné umiestniť do veľkej výšky, avšak v tom prípade dôjde k nežiadúcemu skresleniu obrazu.

Podobne, pre zabezpečenie presnosti polohy rezania, je dôležité minimálne skreslenie kamery. Prirodzene pre uvažovaný typ kamery a cenovú reláciu je nepravdepodobné, aby sa dala táto požiadavka splniť.

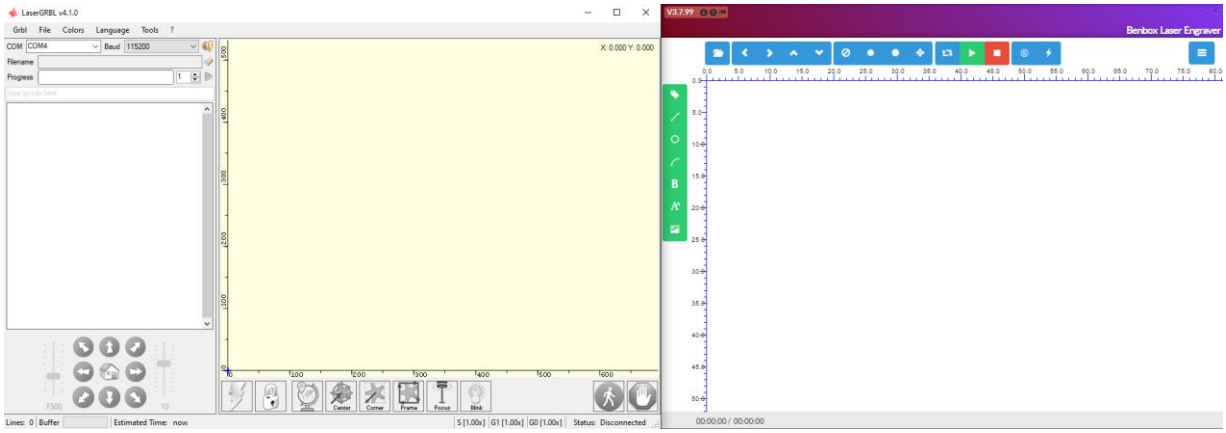
4.1.2 Laserová rezačka

Jedná sa o generickú, bežne dostupnú DIY laserovú rezačku. Rezačka pozostáva z rámu a oporných nôh. Pre pohyb v smere osi *y* sú použité dva krokové motory na opačných stranách rámu a pre pohyb v smere osi *x* postačí jeden krokový motor. Týmto sa zabezpečí pohyb laseru v rámci pracovnej plochy rezačky. Samotný laser je potrebné napájať z externého zdroju, podľa daného výkonu a parametrov. Na ovládanie pohybu krokových motor a eventuálne laseru, je nutné využiť riadiacu dosku (Arduino), ktorá musí zaručiť kontrolu motorov a laseru. Táto doska sa následne pripojí cez USB/micro-USB rozhranie do desktopového počítača/notebooku.



Obrázok 4-1 - Ukážka laserovej rezačky

Existujú dva viac rozšírenejšie firmvéry pre riadiace dosky typu Arduino. Prvým je BenBox[1] a druhým GRBL[2]. Oba prichádzajú so svojimi vlastnými grafickými rozhraniami na rezanie.



Obrázok 4-2 - Grafické rozhrania pre laserové rezačky, GRBL (vľavo) Benbox (vpravo)

[1] https://benboxlaser.us/
[2] https://github.com/grbl/grbl

Je potrebné si uvedomiť rozdiely v jednotlivých firmvéroch. Pre GRBL sa „origin“, počiatok rezacej plochy uvažuje bod [0,0] karteziánskej sústavy a vzdialenosť narastá v kladnom smere ôs. Pri type BenBox je počiatok rovnako v bode [0,0], avšak vzdialenosť od stredu narastá v kladnej osi *x* a v zápornej osi *y*. Už tu vzniká nekompatibilita oboch firmvérov.

Pomocou spomínaných rozhraní sme schopní si nakonfigurovať nastavenia riadiacej dosky na konkrétny typ rezačky, napríklad rozlíšenie kroku (step ~ mm), alebo maximálna možná vzdialenosť pohybu v oboch osách, intenzita laseru a podobne.

Jemný rozdiel je aj v spôsobe komunikácie, respektíve v spätnej väzbe od rezačky. Pri BenBox-e rezačka v zásade spätne nekomunikuje, kdežto GRBL zasiela potvrdzovacie správy a správy o stave po sériovej zbernici. Z tohto hľadiska je GRBL komunikácia robustnejšia a z pohľadu vývoja môže byť prívetivejšia.

```
78  ?<Idle|MPos:0.000,0.000,0.000|FS:0,0>
79  ?<Idle|MPos:0.000,0.000,0.000|FS:0,0|WCO:0.000,0.000,0.000>
80  M5S0                                          <- Príkaz
81  G90
82  G21
83  G1F1000
84  G1X188.1578Y281.4064
85  G4P0
86  M3S950
87  ok                                            <- Odpoveď
88  ok
89  ok
90  ok
91  G4P0
92  G1F750.000000
93  G2X181.9288Y249.6831I-83.8964J0.
94  ok
95  G2X172.6561Y243.4542I-9.2727J3.7874
96  ?<Run|MPos:0.013,0.025,0.000|FS:123,0|Ov:100,100,100>
97  ?<Run|MPos:0.225,0.350,0.000|FS:224,0>
```

Obrázok 4-3 - Príklad komunikácie GRBL

Naopak výhodou pre BenBox je v samotnom rozhraní, ktoré dokáže vygenerovať G-Code z bitmapy, teda z rasterového obrázku a dokáže aj základné operácia nad obrazom. Taktiež vkladanie textu, geometrických primitív alebo aj spôsob rezania (outline).

Spoločná črta je možnosť manipulovať s parametrami lasera – intenzita - a manuálna kontrola pohybu lasera v rámci rezacej plochy.

4.2 Softvérové prostriedky

4.2.1 GUI

V prvom rade sa musí správne zvoliť framework, v ktorom sa bude aplikácia vyvíjať. Malo by sa jednať o najrobustnejšie prostredie so širokým spektrom funkcionalít a možnosťou implementovať všetky potrebné prvky na realizáciu.

Qt predstavuje prostredie na vývoj grafických užívateľských rozhraní a aplikácií, ktoré sú použiteľné na rôznych platformách od Windows, Linux, MacOS či Android, až po embedded zariadenia a webové aplikácie. Qt je založené na princípe widgetov, čo predstavuje kolekciu knižníc na základné grafické ovládacie prvky. Je kompletne písané v C++, avšak podporuje aj iné jazyky ako Python (PyQt), Ruby (QtRuby) C# alebo Java. Taktiež ponúka vlastné vývojové prostredie a nástroje – Qt Creator a Qt Designer. Výhodou je podrobne spracovaná dokumentácia s množstvom príkladov a tutoriálov.

4.2.2 SVG

SVG – scalable vector graphics – je značkovací jazyk popísaný v XML formáte. Umožňuje jednak statické, ale aj dynamické animácie. Je to teda textovo založený formát na opis obrázkov, tým pádom sa dá jednoducho použiť na zobrazovanie vektorovej grafiky v prehliadačoch a iných platformách. Principiálne dokáže popisovať tri druhy grafických objektov :

- Rasterový obraz
- Text
- Grafické primitívy

Výhoda SVG formátu je hneď niekoľko oproti klasickému rasterovému obrázku. Prvou je spôsob popisu pomocou XML formátu, čo predstavuje hierarchický spôsob zápisu pre dáta.

```svg
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   width="200"
   height="200"
   viewBox="0 0 52.916665 52.916665"
   sodipodi:docname="rect.svg">
    <rect
       style="fill:#ffffff;stroke:#000000;stroke-width:0.352518"
       id="rect10"
       width="52.564148"
       height="52.564148"
       x="0.17625892"
       y="0.17625892"
       transform="rotate(45,26,26)"/>
</svg>
```

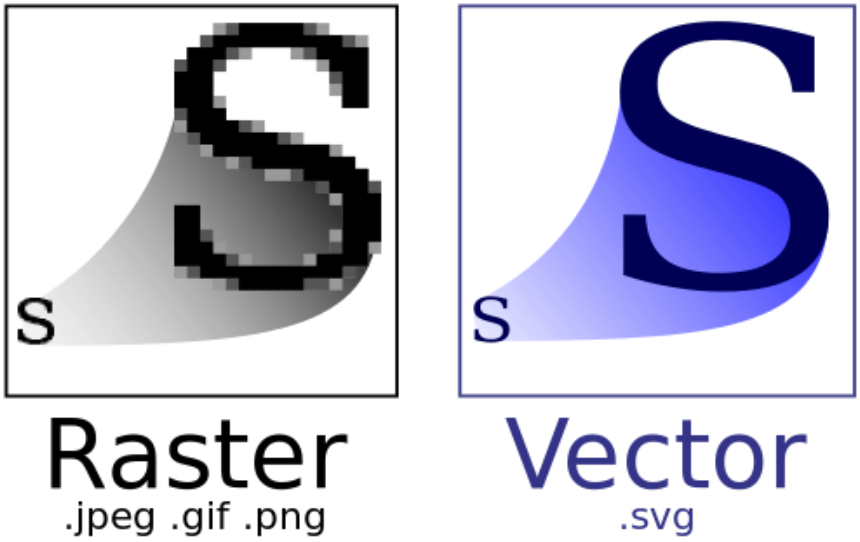
Obrázok 4-4 - Príklad SVG formátu

Štruktúra SVG (XML) formátu je stromová. Súbor začína tzv. prológom a nasleduje koreňový element, na ktorý sa viažu listy (deti). Každý element môže obsahovať textový popis alebo atribút-y.

```
---------------------------------
    <root>
      <child attribute="value" >
        <subchild> TEXT </subchild>
      </child>
    </root>
---------------------------------
```

Pretože ide o štruktúrovaný zápis dát, sme schopní SVG súbor jednoducho rozparsovať a rýchlo pristupovať k jednotlivým uzlom a meniť ich.

Ďalšou výhodou je zobrazovanie. Oproti klasickému rasterovému obrázku je vektorová grafika presnejšia a nestráca kvalitu pri proporčných zmenách (scale, zoom). Jedná sa o zápis grafických entít, ktorých vlastnosti sme schopní programovo (manuálne) meniť. Rovnako sme schopní meniť vlastnosti obrazu, či už ide o farbu, šírku čiary a podobné estetické vlastnosti. V neposlednom rade je podstatná možnosť aplikovať afínnu transformáciu, či už pomocou matice – transfrom matrix - alebo zápisom jednotlivých transformácií – scale, rotate, skew, translate.



Obrázok 4-5 - Rozdiel medzi rasterovým a SVG obrazom[3]

Zásadným rozdielom je rovnako veľkosť SVG súboru oproti klasickým obrazovým formátom. Eventuálne obrovským prínosom SVG formátu je schopnosť jednoduchej konverzie na G-Code pre CNC stroj.

4.2.3 G-Code

G-kód predstavuje sadu inštrukcií na programovanie dráhy pre CNC stroje. Spravidla ide o vektorový pohyb v karteziánskej sústave, kde jednotlivé inštrukcie definujú, o aký pohyb ide a spôsob ich prevedenia. Základné typy inštrukcií sa dajú deliť na „G“ a „M“ inštrukcie. „G“ inštrukcie určujú typ operácie, ktorú má stroj vykonať, napríklad pohyb lineárny alebo kružnicový, spôsob programovania, alebo v akých jednotkách sa komunikuje.

- G0 : Rýchly priamočiary pohyb
- G1 : Lineárny pohyb
- G2 : Pohyb po krivke (kružnici) v zápornom smere
- G3 : Pohyb po krivke (kružnici) v kladnom smere
- G4 : Zastavenie pohybu [ms]
- G21 : Nastavenie jednotiek na *mm*
- G90 : Nastavenie pre absolútne polohovanie

[3] https://www.maketecheasier.com/differences-between-image-file-formats/

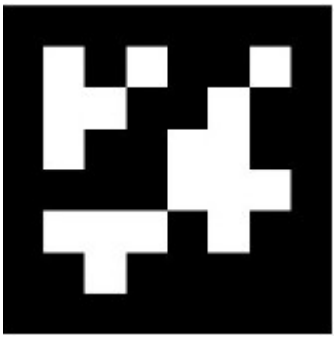
Pre „M“ inštrukcie platí, že ide o pomocné a činné inštrukcie, ako napríklad, ON/OFF nástroja, chladenie alebo iné inštrukcie samotného programu.

- M0 : Stop
- M1 : Sleep
- M3 : Laser ON
- M5 : Laser OFF

4.2.4 OpenCV

OpenCV je open-source knižnica používaná v oblasti počítačového videnia, teda obsahuje obrovské množstvo algoritmov na riešenie úloh real-time v tejto oblasti. Knižnica má modulárnu štruktúru a obsahuje zdieľané a statické knižnice, ktoré zahrňujú základné funkcionality a moduly až po extra nadstavbové moduly. Knižnica je primárne písaná a udržovaná v jazyku C++ (/C), taktiež sú podporované jazyky ako Python, Java alebo Matlab. Dôležitou vlastnosťou je cross-platfromová implementácia a je voľná pre (ne)komerčné použitia.

Aruco markery - Sú to binárne fiduciály, ktoré sa používajú v rôznych aplikáciách v počítačovom videní : navigácia robotov, rozšírená realita a podobne. Marker pozostáva z čierneho okraja a vnútorného, ktorý predstavuje binárnu maticu markeru. Taktiež každý marker disponuje svojím vlastným ID. Obyčajne markery obsahujú binárnu maticu o rozmeroch 4x4 až 7x7.

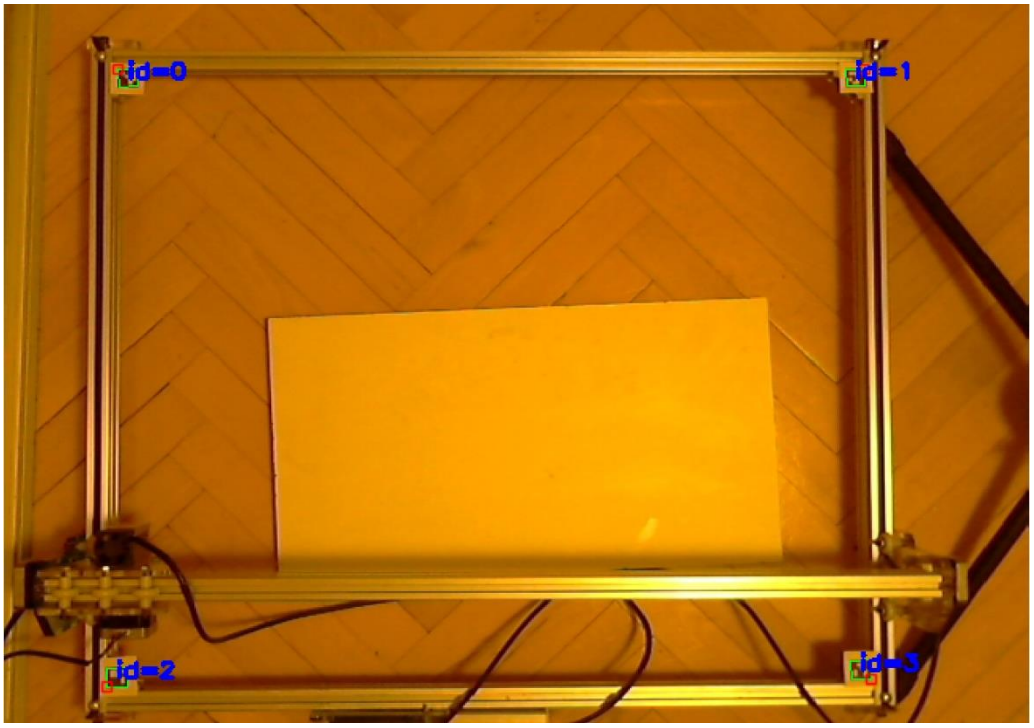


Obrázok 4-6 - Príklad ArucoMarkeru (7x7)

Tzv. slovníky obsahujú vlastnosti každého generovateľného markeru. Slovníky určujú počet markerov a ich rozmer (binárnu maticu, dĺžku hrany v px). Knižnica aruco obsahuje už predpripravené slovníky, avšak je možné si aj vytvoriť vlastné, ak je to špecificky potrebné.

Generovanie týchto markerov je robené programovo, kde sa určia parametre – binárna matica, dĺžka hrany, veľkosť čierneho okraja a konkrétny slovník. Následne generované obrázky je potrebné vytlačiť a umiestniť.

Výstupom procesu detekcie týchto markerov je ich unikátne ID a pozícia markeru v obraze. Tento proces pozostáva z viacerých krokov a úspešnosť detekcie závisí na dostatočnom rozlíšení kamery a aby markery boli na dostatočne kontrastnom pozadí, inak sa znižuje pravdepodobnosť detekcie.



Obrázok 4-7 - Príklad detekcie markerov

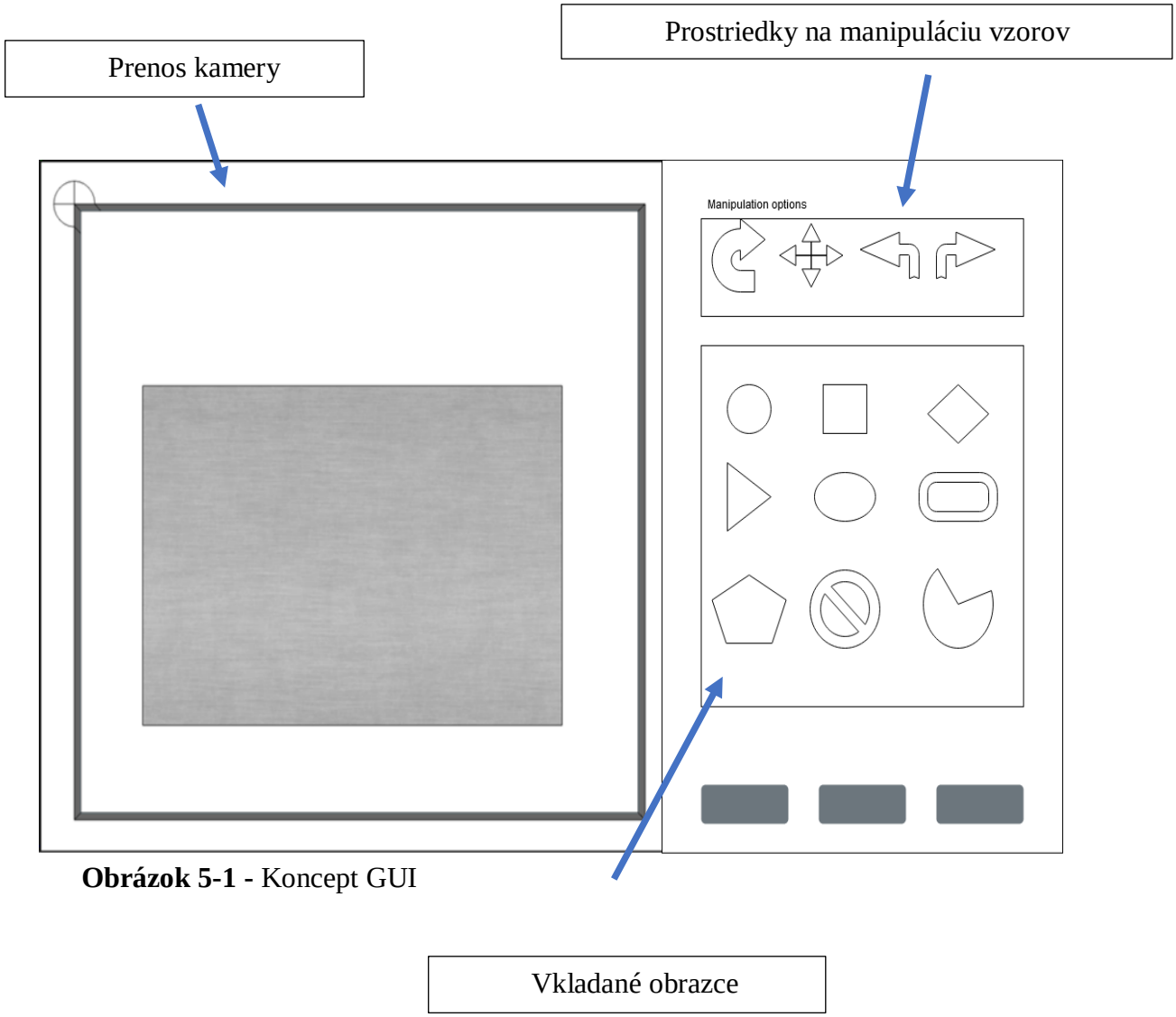
5 IMPLEMENTÁCIA

V tejto sekcii rozoberám konkrétne implementačné prostriedky použité pri vývoji práce alebo aplikácie. Od konkrétnych hardvérových prostriedkov, softvérových nástrojov až po spôsoby implementácie a prekážky, ktoré vznikli počas procesu vývoja.

Úlohou bolo vytvoriť grafické užívateľské rozhranie, pre danú laserovú rezačku, s prvkami rozšírenej reality – vývoj aplikácie.

Koncept GUI je zložený z :

- Prenosu kamery
- Vkladaných obrazcov
- Možnosťou manipulácie s obrazcami
- Spustenie procesu rezania



Obrázok 5-1 - Koncept GUI

Prenos kamery zabezpečí real-time pohľad na rezačku, tým je možné realizovať prvky rozšírenej reality ako trekovanie a zároveň máme okamžitú informáciu o pozícii materiálu.

Vkladané obrazce sú typu SVG, ktoré je možné jednoducho konvertovať na G-Code. Následne tieto obrazce môže užívateľ vkladať na plochu prenosu kamery a manipulovať – rotácia, translácia.

Posledným krokom je samotný proces rezania materiálu, ktorý spočíva z prenosu G-Code-u do riadiacej dosky rezačky.

5.1 Použitý hardvér

Aplikácia bola vyvíjaná pomocou webkamery, Logitech HD Webcam C270[4]. Úmyslom bolo vybrať bežne dostupnú kameru na trhu s prijateľnou cenou. Spôsob vývoja sa teda zameral na zvládnutie kamery s horšími parametrami.

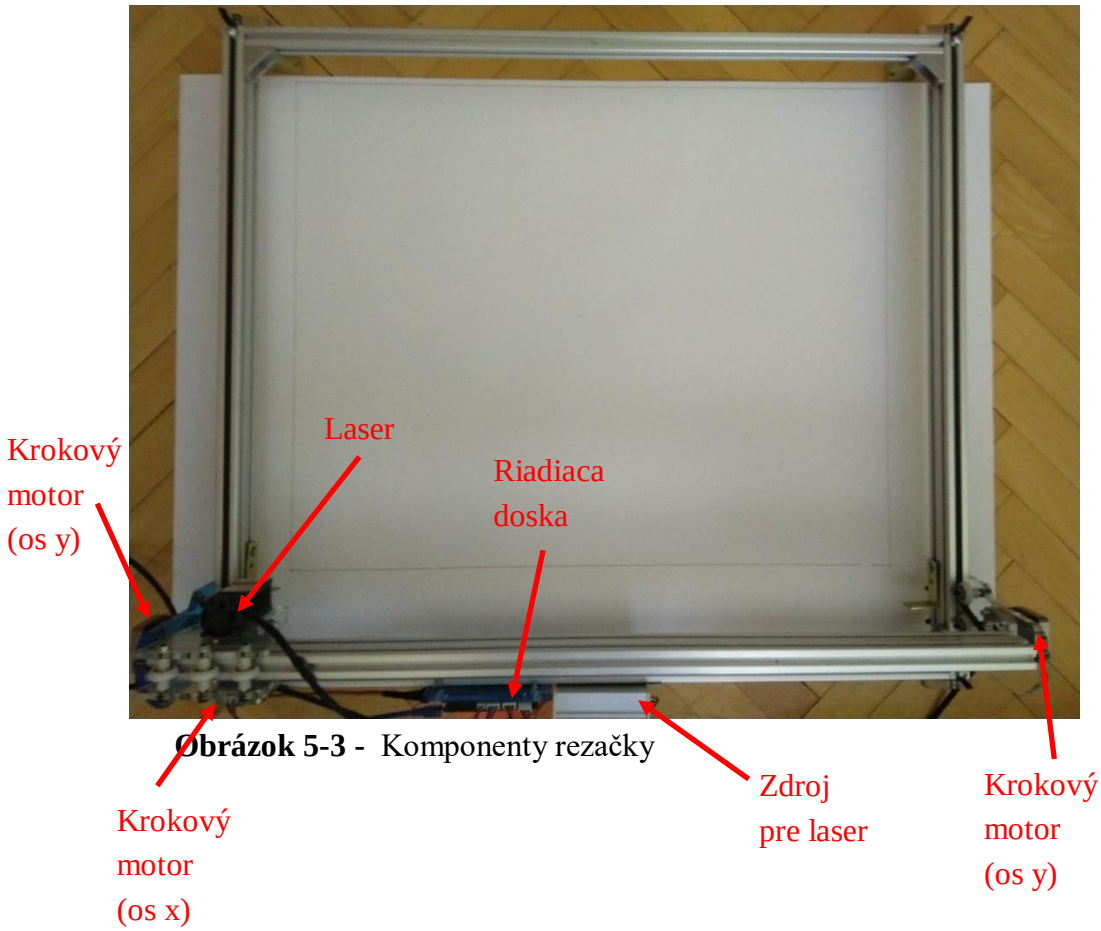
Obrázok 5-2 - Webkamera Logitech C270

Tabuľka 5-1 - Parametre kamery Logitech C270

Rozhranie	High Speed USB 2.0
FOV	60°
Rozlíšenie	1280x960 (1.2MP)
FPS	30 (@640x480)
Ohnisková vzdialenosť	4mm

[4] https://support.logi.com/hc/en-150/articles/360023462093-Logitech-HD-Webcam-C270-Technical-Specifications

Použitá laserová rezačka pozostáva z nosnej konštrukcie o vonkajších rozmeroch 722x632 mm, na ktorej je umiestnená pohyblivá sústava lasera. Na pohyb slúžia krokové motory[5], dva pre *os y* a jeden pre *osu x*. Na ovládanie motorov slúži riadiaca doska.



Obrázok 5-3 - Komponenty rezačky

Tabuľka 5-2 – Parametre riadiacej dosky

Kód produktu	SKU577980
Rozhranie	mini-USB
Veľkosť	97x54mm
Výkonové požiadavky	>12 V , 2 A
Podpora krokových motorov	< 2 A, dvojfázové, štvorvodičové
Mikrokontrolér	ATmega328P[6]
Doska	Arduino Nano[7]

[5] https://components101.com/motors/nema17-stepper-motor
[6] https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
[7] https://store.arduino.cc/arduino-nano

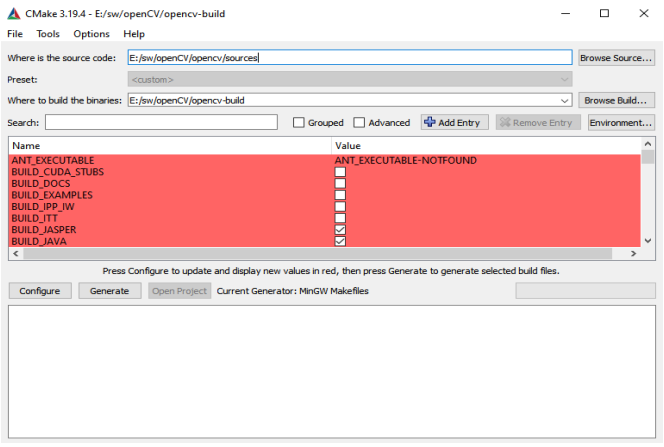
Doska je napájaná pomocou laboratórneho zdroja Basetech BT-305[8].

Obrázok 5-4 - Basetech BT-305 laboratórny zdroj

5.2 Konfigurácia softvéru

Doska bola dodaná už s nahratým firmvérom BenBox. Ako už bolo spomínané, lepšia alternatíva na vývoj sa ukázal byť firmvér od GRBL. Na nahratie GRBL firmvéru bol použitý softvéru Arduino 1.8.13[9]. Následne boli parametre rezačky nakonfigurované pomocou GRBL rozhrania (príloha).

Qt ako také podporuje množstvo kompilátorov v závislosti na platforme. Pre Windows 10 sú podporované 32/64-bit MSVC (2019, 2017, 2015) a MinGW 8.1. Voľba kompilátora bola určená na základe spojenia Qt a OpenCV[10]. Táto väzba bola teda vytvorená pre MinGW 8.1 64-bit, pomocou CMake, kde sa skompilovala knižnica OpenCV (vytvorenie objektov) a rovnako sa vytvorila spomínaná väzba.



Obrázok 5-5 - Ukážka CMake

Týmto spôsobom sa umožnilo používanie API OpenCV priamo v Qt.

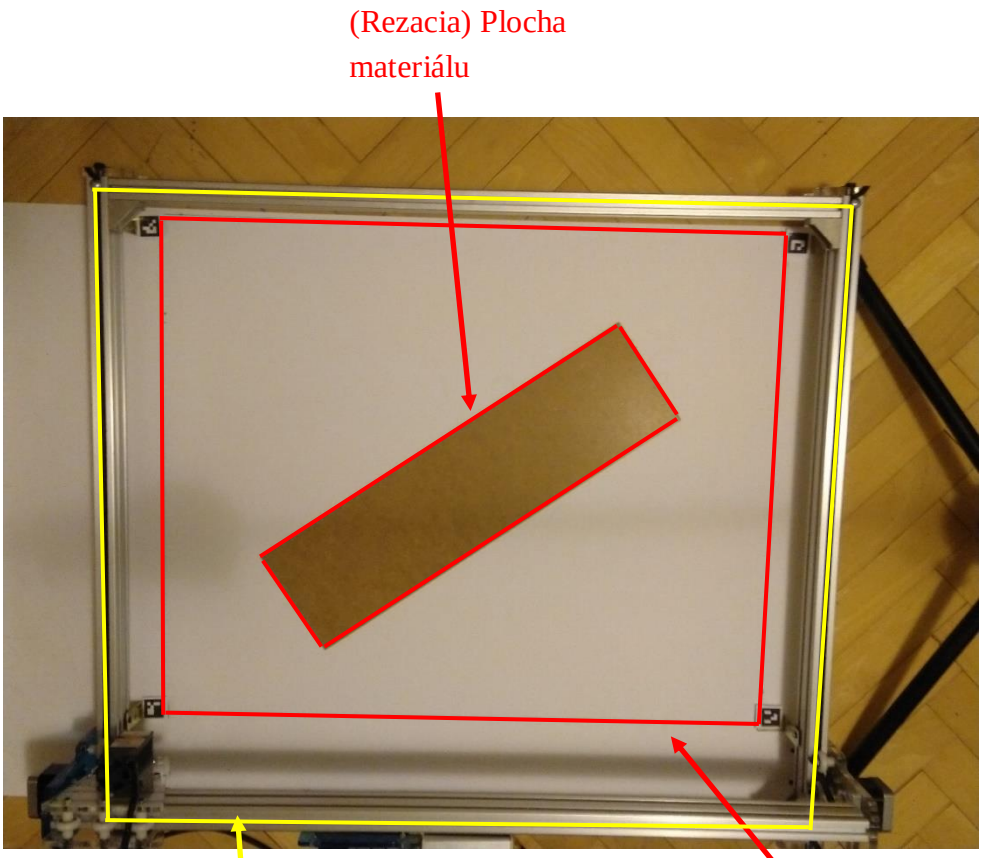
[8] https://www.conrad.sk/p/basetech-bt-305-laboratorny-zdroj-s-nastavitelnym-napatim-0-30-vdc-0-5-a-150-w-pocet-vystupov-1-x-513812
[9] https://www.arduino.cc/en/software
[10] https://wiki.qt.io/How_to_setup_Qt_and_openCV_on_Windows

5.3 Koncepcia riešenia

Súradnicový systém laserovej rezačky je definovaný svojou dĺžkou a šírkou, avšak skutočná rezacia plocha je menšia. Táto plocha je definovaná voľnosťou pohybu samotného lasera, čiže schopnosť pohybu v oboch osách. Môžeme si to predstaviť ako obdĺžnik (jeho obsah), ktorý je podmnožinou plochy o rozmeroch rezačky.

Termíny :

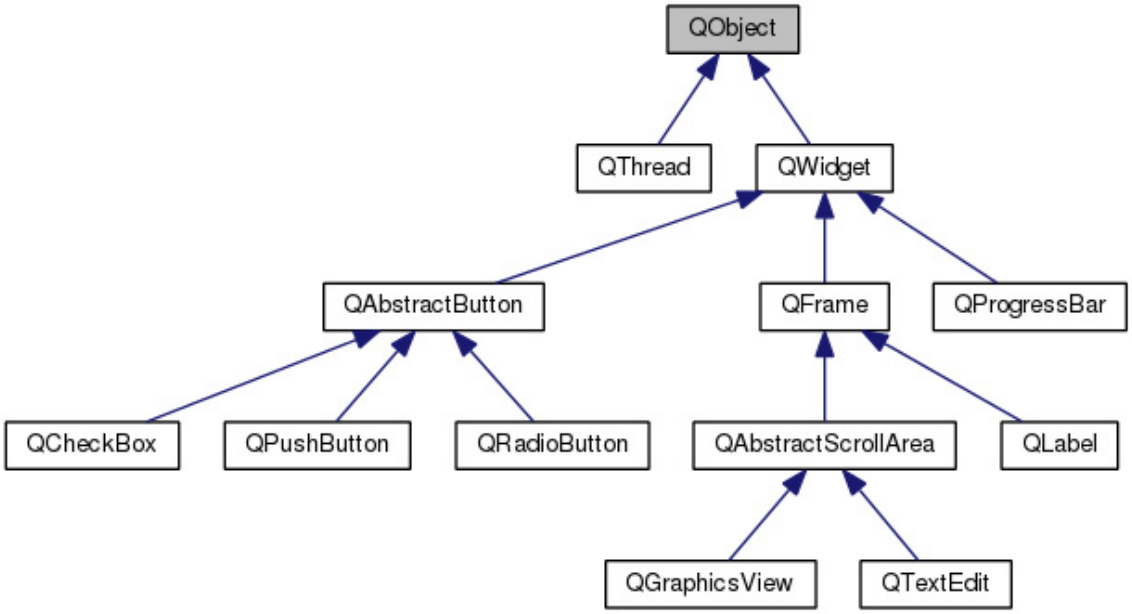
- Rezačka – rozmery rezačky, jej rámec
- Rezacia plocha – vnútorná plocha na rezanie definovaná voľnosťou pohybu lasera
- (Rezacia) Plocha materiálu – plocha definovaná materiálom, na ktorý je možné rezať



Obrázok 5-6 - Definovanie pojmov pre rezačku

5.3.1 Problém s modifikovaním SVG

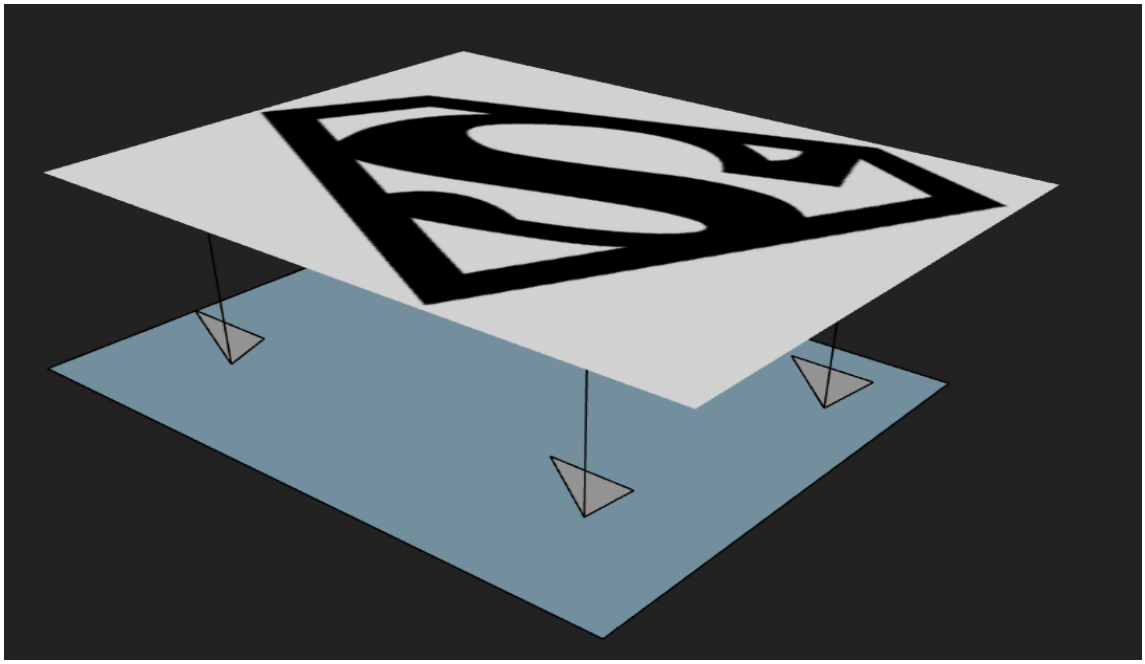
Qt pri zobrazovaní pracuje v triedach tzv. widgetov, čo sa dá rozumieť ako zobrazovacia plocha. Každá táto plocha je unikátna trieda so špecifickými vlastnosťami a možnosťami.



Obrázok 5-7 - Názorná štruktúra Qt tried[11]

Prirodzene toto je len zlomok toho, čo Qt ponúka v rámci samostatných tried. Podstatné je, že pre zobrazenie akéhokoľvek obrázka sa použije bitmapa. Teda už len z toho plynie, že sa ničí celková myšlienka vektorového zobrazovania.

[11] https://wiki.qt.io/Qt_for_Beginners



Obrázok 5-8 - Princíp zobrazovania obrázkov v Qt

Síce Qt chce byť, čo najrobustnejšia platforma a preto sa snaží podporovať čo najviac možností, natívne nepodporuje priame alebo automatizované manipulovanie so súbormi typu SVG. Priamo sa podporuje iba ich zobrazovanie a ukladanie. Existuje špeciálny widget – QSvgWidget – ktorý slúži na zobrazovanie SVG formátu, avšak stále len v kontexte zobrazovania.

Ďalšia možnosť pre prácu s grafickými položkami je QGraphicsItem a triedy, ktoré s ním pracujú. V tomto prípade sa dá prirodzenejšie pracovať s grafickými inštanciami. Jednoducho sa dajú meniť zobrazované obrazy a programátor nie je prakticky obmedzený zobrazovacou plochou. Avšak všetky tieto zmeny sa priamo nepremietnu, pri ukladaní, do výsledného súboru SVG.

Pri ukladaní na SVG súbor, existuje trieda QSvgGenerator, ktorá dokáže generovať SVG súbor. Lenže správne generovanie SVG formátu dokáže iba z geometrických primitív, ktoré sa „nakreslia“ priamo v Qt, inak sa generuje SVG z bitmapy, čo vlastne nie je vektorový formát.

Obrázok 5-9 - Ukážka referencie obrázka v Svg súbore

Vygenerované SVG dostane referenciu na obrázok ako na zdroj, čiže sa nevygeneruje SVG správne, ale iba obrázok (bitmapa), ktorý sa tvári ako SVG. Čiže tu plynie prvá prekážka – Qt nedokáže priamo manipulovať s SVG súborom, meniť jeho vlastnosti.

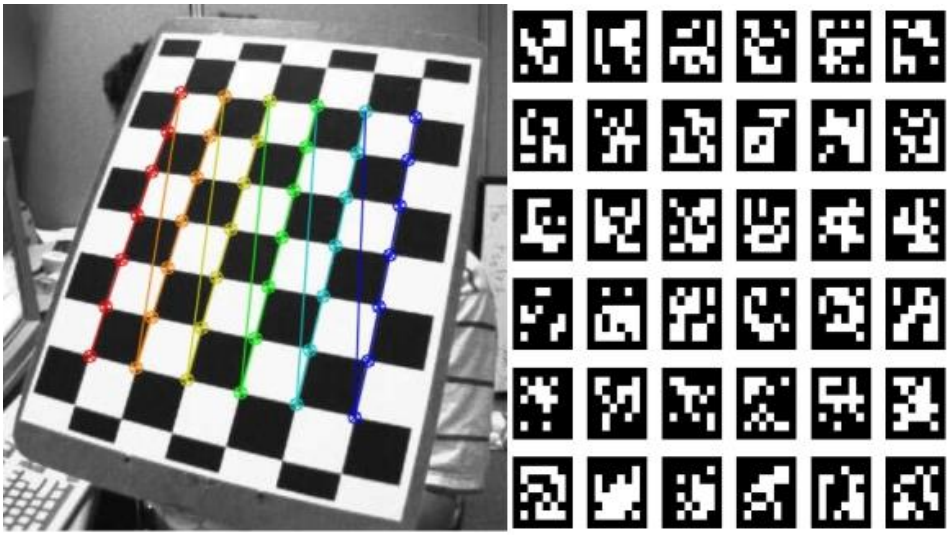
Keby som to mal zhrnúť, Qt podporuje iba zobrazovanie a veľmi obmedzene, ukladanie SVG formátu. Priamo nepodporuje zmenu vlastností. Pre korektné vytváranie SVG súborov by sa musel vektorový obraz, doslova napísať v XML alebo inom jazyku.

Manipulácia obrazu (rasterového) v samotnom Qt je možná, či už cez transformácie alebo priame zmeny veľkosti zobrazovacej plochy a podobne. Týmto spôsobom je možné si istým spôsobom zapamätať zmeny nad konkrétnym obrazcom a následne ich aplikovať na dané SVG priamo v XML súbore.

5.3.2 Riešenie s kalibráciou kamery

Prvotný koncept riešenia sa snažil umiestniť ArucoMarkery na rohy rezačky (vnútorné). Tým by sme boli schopní nájsť súradnicový systém rezačky a markery sú pevne spojené s rezačkou, čo je užívateľsky prívetivejšie a jednoduchšie na použitie. Následne by prebehla kalibrácia kamery, potom by sa našiel materiál a určila vzájomná poloha rezačky a materiálu. Tým by sme sa dostali zo súradnicového systému rezačky na súradnicový systém materiálu. Posledným krokom by bolo určiť vzťah px~mm na prepočet umiestneného obrazca v GUI.

Problém riešenia spočíva v samotnej kalibrácii kamery. Kalibráciu kamery je možné realizovať pomocou šachovnice z OpenCV alebo „šachovnice“ zloženej z ArucoMarkerov, respektíve ChArucoMarkerov, tzv. „Board“ – doska.



Obrázok 5-10 - Kalibračné objekty, šachovnica (vľavo), ArucoBoard (vpravo)

Kalibrovaním kamery, zistením interných parametrov kamery, by bolo možné prenos kamery korigovať, korigovať jej skreslenie. Týmto spôsobom by sa prenos kamery upresnil, respektíve poloha vkladaných obrazcov by bola presnejšia.

Na kalibráciu kamery je avšak nutná množina kalibračných snímkov, z ktorých sa určí matica kamery. Nie je možné určiť túto maticu len z jedného snímku. Keď sa pozrieme na koncept úlohy, v princípe si užívateľ „namieri“ kameru na rezačku a mieni začať využívať rezačku. Nemusí byť schopný zvládnuť značne náročnú operáciu kalibrácie kamery. Nieto ešte, aby ju realizoval po každej zmene pozície kamery voči rezačke.

Bez kalibrácie kamery sa tak zvýši chyba pri ukladaní obrazcov a následnom vyrezávaní.

5.3.3 Riešenie bez kalibrácie kamery

V princípe nás nezaujíma, kde sa nachádza rezačka, ako skôr to, kde sa nachádza rezacia plocha. Toto je náš referenčný súradnicový systém, na ktorý sa bude vzťahovať celé riešenie. Na definovanie rezacej plochy je teda potrebné definovať voľnosť pohybu lasera, čo sa zistilo jeho jednoduchou manipuláciou pomocou GRBL rozhrania a vyznačením tejto plochy laserom.

Po definovaní rezacej plochy musíme túto plochu byť schopní nájsť v priestore, respektíve v obraze. Toto práve zaručia ArucoMarkery, rozmiestnené na rohy rezacej plochy.

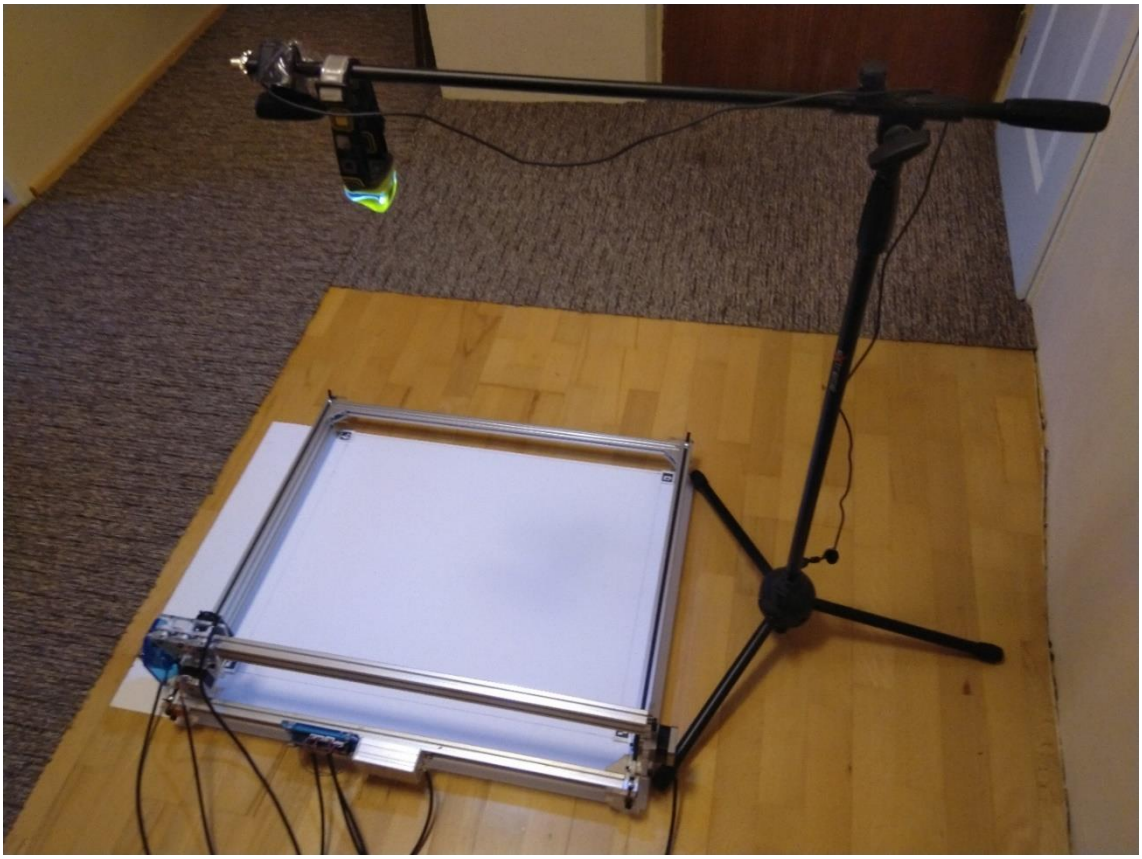
Obrázok 5-11 - Spôsob umiestnenia ArucoMarkerov na rohy rezacej plochy

Podstatnou skutočnosťou, ktorú je nutné zaistiť v tomto prípade, je, aby ArucoMarkery boli pevne spojené s rezačkou. Takto sa zaručí, že rezacia plocha vyhradená ArucoMarkermi pevne zodpovedá skutočnej rezacej ploche rezačky a nedôjde k zväčšení chyby pri rezaní. Ak by ArucoMarkery boli umiestnené na zemi, nohy rezačky by sa museli umiestniť na presne definované pozície a nemohlo by dôjsť k zmene jej polohy.

Na polohovanie ArucoMarkerov slúžia externe primontované kovanie v tvare „L“. (uholníková spojka). Rozmery boli adekvátne upravené. Približne 25x40x20 mm.

Obrázok 5-12 - Nábytkový uholník

5.3.4 Koncepcia scény



Obrázok 5-13 - Pohľad na scénu

Scéna pozostáva z kamery, ktorá zvrchu sníma rezačku, tak, aby boli zachytené všetky ArucoMarkery. Tým sa získa rezacia plocha, definovaná ArucoMarkermi na ďalšie spracovanie.

5.4 Softvérová implementácia

Ako už bolo spomínané, celé softvérové riešenie je uskutočnené v Qt pomocou knižnice OpenCV kompletne písané v C++.

Na konverziu na G-Code a jeho následné zasielanie bol vytvorený pomocný Python skript, respektíve súbor .exe. Tento skript čerpá z package-u *svg-to-gcode-1.4.1*[12].

[12] https://pypi.org/project/svg-to-gcode/1.4.1/

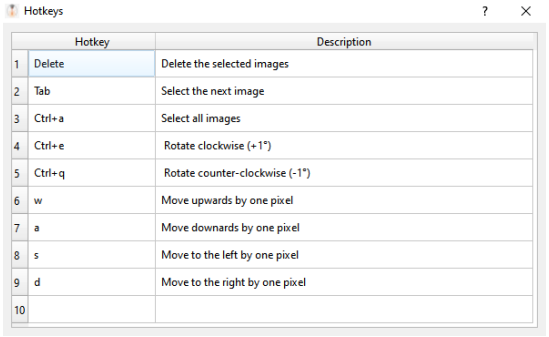
Vygenerované .exe vyžaduje tri argumenty (-c -s -p) v ľubovoľnom poradí :

- -c/--comport [COMx]
- -s/--svgfile [absolútna cesta k .svg súboru]
- -p/--gfile [absolútna cesta na .gcode]
- -h/--help

Kód je rozdelený do štyroch tried:

1. Trieda „DragWidget“ zabezpečuje Drag&Drop mechanizmus medzi zdrojovými obrázkami a plochou zobrazenia kamery

2. Trieda „Hotkeys“, ktorá zobrazí do nového okna tabuľku klávesových skratiek



Obrázok 5-14 - Okno klávesových skratiek

3. Trieda „SvgLabel“, vlastný typ triedy (dedí z QLabel), ktorý má na starosť zobrazované obrazce a manipuláciu s nimi.

4. Hlavná trieda „LaserCutter“ spája všetky svoje a funkcionality ostatných tried do finálneho celku.

Posledným členom je pomocný namespace „FileHandler“ pre narábanie so súbormi.

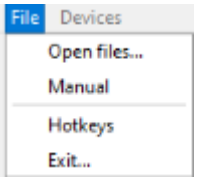
Pomocné súbory zahŕňajú *User Manual* pre orientovanie a následnosť krokov v aplikácii, tak, aby užívateľ bol schopný správne porozumieť celému procesu, konfiguračný súbor *config.xml*, z ktorého sa inicializujú niektoré parametre pre aplikáciu a súbor *surface_final.svg*, ktorý je prázdny SVG súbor a do ktorého sa pridávajú geometrické primitívy obrazcov, ktoré sa budú rezať.

5.4.1 Užívateľské rozhranie

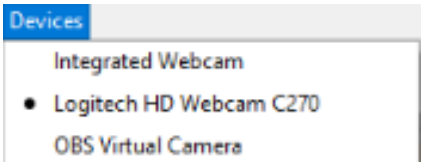
Užívateľské rozhranie je možné rozdeliť do štyroch častí :

Obrázok 5-15 - Ukážka aplikácie

[1] Možnosť *File*, kde užívateľ je schopný zvoliť zdrojový adresár SVG súborov. Rovnako si môže zobraziť *User Manual* a *Hotkeys* tabuľku.
Možnosť *Devices*, užívateľ si môže zvoliť jednu z dostupných kamier pripojených v systéme



Obrázok 5-16 - Files menu



Obrázok 5-17 - Devices menu

[2] Zdrojové obrázky načítané zo zdrojového adresára. Absolútna cesta k adresáru je uložená do súboru *config.xml*, z ktorého je defaultne načítaná pri inicializácii aplikácie.

[3] Prenos kamery, pred aj po použití perspektívnej transformácie.

[4] Aktívna časť, zväčša závislá na inpute používateľa.

- Možnosť zvoliť *COM port*, na ktorý je pripojená rezačka.
- Určiť parametre rezacej plochy (parametre sú uložené a načítavané z *config.xml* súuboru, ak ich užívateľ nezmení)
- *Rotačné točidlo*, používaný na rotáciu vkladaných obrazcov
- *Posuvný jazdec*, ktorým sa určuje správny *prah* na kontúry materiálu a jeho následné trekovanie
- *Start/Reset* tlačítko na inicializáciu, respektíve reštart aplikácie (interný reštart)
- *Clear Workspace*, vymazanie obrazcov z pracovnej plochy (prenos kamery)
- *Cut*, započatie konečného procesu rezania
- Informácie o momentálne zvolenom obrazci, *pozícia (x,y)*, *rotácia*. Pozíciu je možné zobraziť v milimetroch alebo pixeloch. (Pozícia je uvažovaná vzhľadom na ľavý horný roh prenosu kamery)

5.4.2 Použité nástroje OpenCV

Ako už bolo spomínané, na zistenie referenčnej plochy, rezacej plochy, rezačky sú použité ArucoMarkery, ktoré sú umiestnené na rohoch rezacej plochy.

Tabuľka 5-3 - Parametre použitých ArucoMarkerov

Bitové slovo	Dĺžka hrany	Slovník
4x4	70px	4x4x250

Parametre markerov boli určené experimentálne.

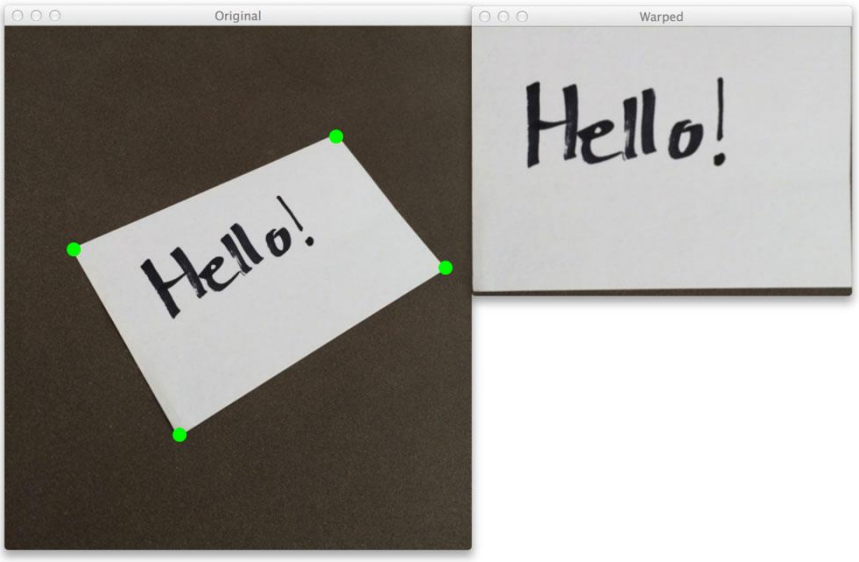
Obrázok 5-18 - Detekcia Aruco markerov 4x4 , v ľavo hore **40p**, vpravo hore **50p**, vľavo dole **60p**, vpravo dole **70p**

Vychádzalo sa z nárokov na kameru, keďže aplikácia bola vyvíjaná pomocou bežnej webkamery, detekovanie markerov by malo byť čo najjednoduchšie. Čím menšie je bitové slovo, tým je menej náročná detekcia – väčšia pravdepodobnosť, že dôjde k detekcii. Dĺžka hrany sa určila experimentálne pre dĺžky hrán 40,50,60,70 pixelov. Pri 70 pixeloch sa detekcia správala najkonzistentnejšie. Opäť je potrebné zdôrazniť, že markery by mali byť umiestnené na dostatočne kontrastnom pozadí, aby vôbec došlo k ich detekcii.

Proces detekcie zastrešuje metóda *LaserCutter::CameraMarkerDetection()*[13].Detekcia ArucoMarkerov spočíva v udaní rohov týchto markerov. Roh s indexom 0 je na obrázku hore zvýraznení červeným štvorcom, a ten je referenčný (pre každý marker) pre následnú perspektívnu transformáciu. Preto je potrebné, aby tieto rohy boli umiestnené v rohoch rezacej plochy.

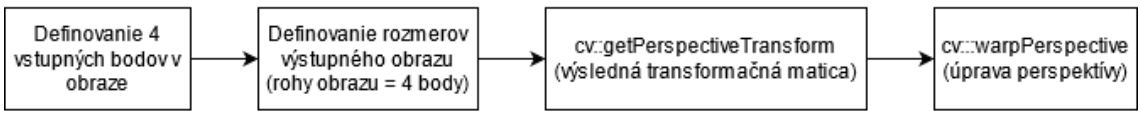
Z prenosu kamery je nutné zistiť štyri body, ktoré definujú rezaciu plochu. Následne môžeme túto plochu extrahovať a zobraziť. Tento proces využíva tzv. 4–bodovú perspektívnu transformáciu.

[13] https://docs.opencv.org/3.4/d5/dae/tutorial_aruco_detection.html



Obrázok 5-19 - Ukážka 4-bodovej perspektívnej transformácie[14]

Princíp spočíva v určení štyroch bodov, ktoré definujú bounding box transformovanej oblasti. Následne sa určí vzťah bodov z obrazu a bodov finálneho obrazu (prakticky štyri rohy výsledného obrazu – podľa požadovanej šírky a výšky). Týmto získame transformačnú maticu a „skrútením“ (warping) perspektívy dostaneme rovnobežný/kolmý extrahovaný obraz.



Obrázok 5-20 - Následnosť procesu perspektívnej transformácie

Výsledkom je stotožnenie súradnicového systému rezacej plochy a súradnicového systému zobrazovacej plochy (zobrazenie v aplikácii). Užívateľ vníma, že vkladá obrazce priamo na rezaciu plochu, rovnako sa týmto spôsobom maximalizuje plocha v aplikácii na vkladanie obrazcov.

[14] https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/

Obrázok 5-21 - Prenos kamery pred (vľavo), prenos kamery po transformácii

5.4.3 Spôsob modifikácie SVG

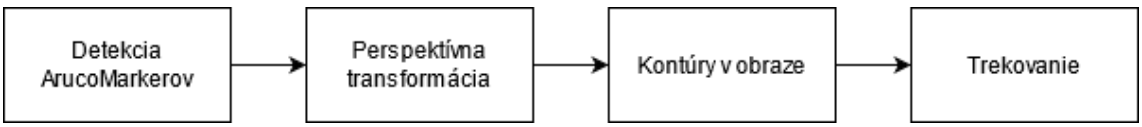
Ako už bolo spomínané, modifikovať SVG je nemožné pomocou samotného Qt. Pre zmeny nad SVG je nutné parsovať tieto súbory a následne ich modifikovať. Pomocou transformácií je možné meniť SVG obrazce. Jedná sa o afínne transformácie zapisované pomocou atribútu *transform* – hodnota atribútu môže pozostávať z konkrétnych transformácií, *scale, translate, rotate, skew* alebo transformačnej matice *matrix*.

```
<path transform="matrix(0.587785,0.809017,-0.809017,0.587785,258.813,271.501)"
```

Zapísaním tejto transformácie sa modifikuje daná primitíva SVG (path, circ, rect...). Konečné SVG sa vygeneruje pridávaním jednotlivých primitív všetkých obrazcov do spoločného súboru (*surface_final.svg*).

5.4.4 Inicializácia aplikácie

Po stlačení tlačítka *Start* sa inicializujú parametre z *config.xml*, následne sa použije reťaz spracovania obrazu.



Obrázok 5-22 - Reťaz spracovania obrazu

Detekujú sa markery, čím sa definujú štyri body ako vstup do perspektívnej transformácie. Potom sa transformuje obraz a nájdu sa kontúry v obraze. Pomocou kontúr, respektíve bounding boxu, sa umožní trekovanie materiálu na základe zmeny polohy stredu tohto bounding boxu.

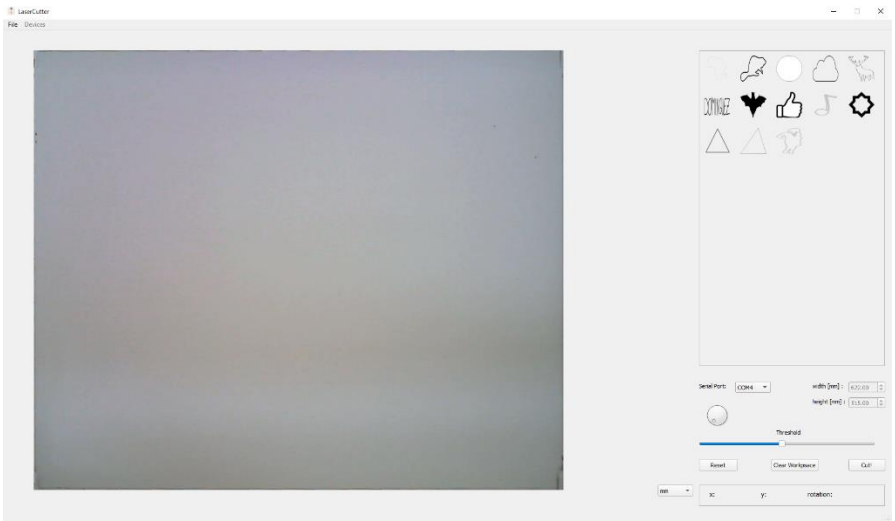
Po detekovaní markerov užívateľ nesmie pohnúť kamerou alebo rezačkou. Detekcia prebieha iba pri inicializácii, neustála detekcia sa ukázala ako časovo náročná.

Obrázok 5-23 - Aplikácia pred štartom

Ďalším krokom je vytvorenie dočasných adresárov *Temp_img_source*, do ktorého sa skopírujú zdrojové obrázky (SVG) a *Temp_img*, kde sa budú ukladať vkladané obrazce ako aj finálny obrazec SVG, ktorý sa použije na vytvorenie G-Codu.

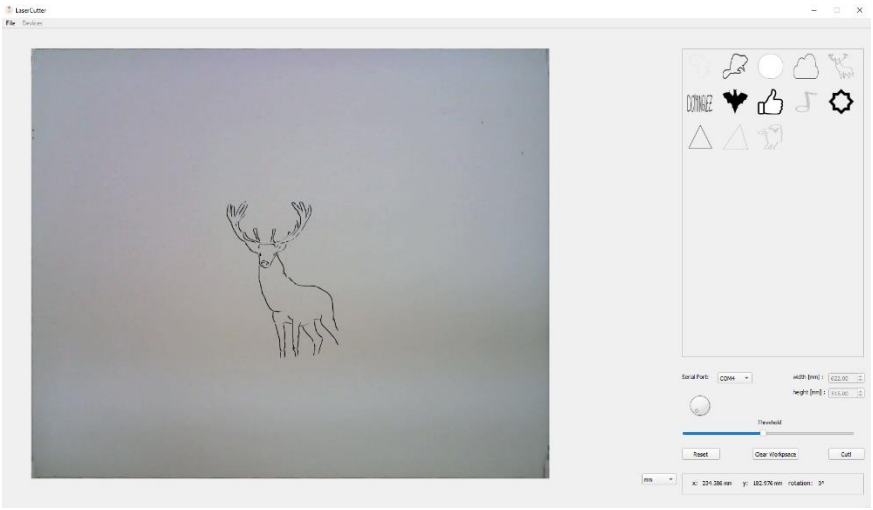
5.4.5 Priebeh aplikácie

Po inicializácii je možné vkladať obrazce na transformovaný prenos kamery



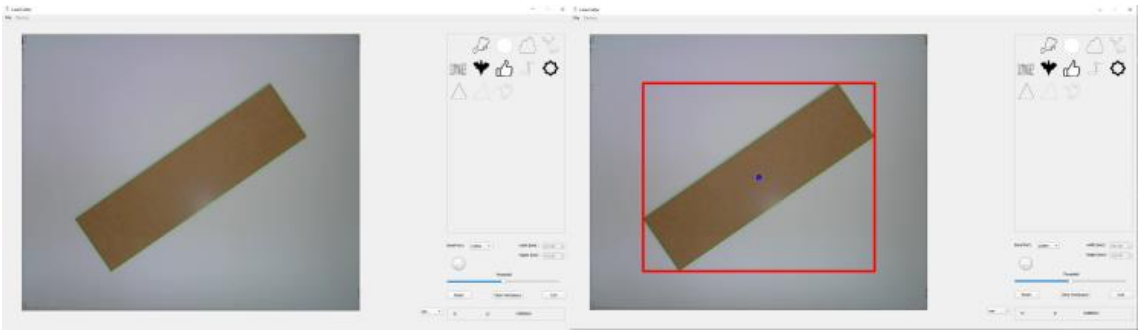
Obrázok 5-24 - Aplikácia po štarte

V tom momente sa určí vzťah **px~mm** v oboch osiach. Obrazce vpravo, ktoré je možné vkladať, sú zobrazené v zmenšenej veľkosti (je možné vložiť 60 obrazcov). Po položení obrazca na plochu prenosu kamery sa obrazec zväčší do adekvátnej veľkosti podľa vzťahov px~mm. Týmto spôsobom sa obrazec zobrazí v skutočnej veľkosti. Následne je možná manipulácia s obrazcom – presun, rotácia.



Obrázok 5-25 - Vloženie obrazca

.Trekovanie materiálu spočíva v nájdení kontúr materiálu v obraze. Bounding box kontúr, respektíve jeho stred, je referenčný bod, na ktorý sa vzťahuje trekovanie a teda obrazce sa posunú na základe zmeny pozície tohto stredu. Na konzistentnejšie správanie je vhodné použiť priame osvetlenie na rezaciu plochu.



Obrázok 5-26 - Kontúry materiálu (vľavo), bounding box (+stred) (vpravo)

Ďalším krokom je samotný proces rezania. Na správnosť rezania sa musia obrazce rezať na ploche materiálu a nie mimo. To zaručí metóda *LaserCutter::CheckSvgsPositions()*. Mechanizmus je založený na segmentácii obrazu. Obraz kamery sa adaptívne prahuje, vzniknú tak dva obrazce – inverzný a neinverzný –

v zásade, kde materiál je biely a pozadie čierne a naopak. Rovnaký proces prebehne s plochou pre vkladanie obrazcov (vrstva nad prenosom kamery).

Následne...

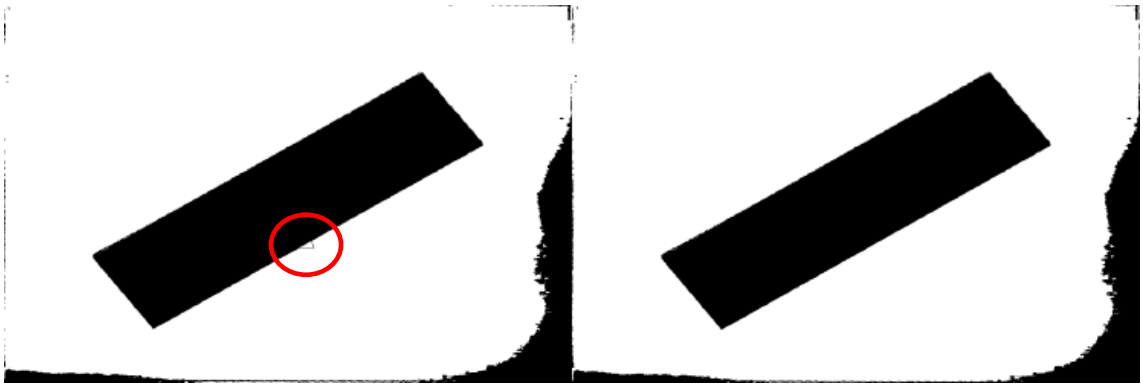
```
white_label = ~white_label;

cv::Mat aux_white = camera_frame_white & white_label;
cv::Mat aux_black = camera_frame_black & black_label;

if(MatIsEqual(camera_frame_white, aux_white) || MatIsEqual(camera_frame_black, aux_black))
    aux = true;
else
    aux = false;
```

Obrázok 5-27 – Časť kódu, či sa obraz nachádza na ploche materiálu

Ak po operácii AND obrazov sa aspoň jeden rovná pôvodnému, obrazec sa nachádza na ploche materiálu.



Obrázok 5-28 - Ukážka zisťovanie správnej pozície obrazcov

Segmentáciu je potrebné vyhodnocovať aj v inverznej forme, aby sa zaistilo zachytenie aj svetlejších/tmavších obrazcov ako je materiál.

Na koniec sa použije vygenerovaný program *SVG2GcodeSend.exe* , ktorý skonvertuje finálne .svg na G-Code, ktorý sa pošle rezačke.

5.4.6 Podmienky použita

Na správne fungovanie aplikácie je nutné zaistiť homogénne pozadie svetlej farby, respektíve podložky rezačky. Zaistí sa tým detekcia kontúr v obraze, rovnako aj korektnosť procesu zisťovania správnosti uloženia obrazcov na ploche materiálu. Predpokladá sa, že materiál je zvyčajne tmavšej farby.

Súbory SVG nesmú obsahovať transformácie – *transform matrix*, *transform* . Kód je koncipovaný tak, že dopĺňa transformácie, čiže transformácie už prítomné v súbore sa vymažú. Súbor SVG taktiež musí byť v jednotkách milimetroch (pre správne

vzdialenosti v G-Code) a rovnako nesmie obsahovať geometrickú primitívu <*rect*> , pretože python konvertor nepodporuje konverziu tohto geometrického typu.

Pre správnosť funkcie trekovania sa na rezacej ploche v jeden moment môže nachádzať iba jeden materiál. Inak by sa vyhodnocovala zmena polohy stredu nejakého materiálu na rezacej ploche, čo by nekorešpondovalo s uvažovaným stredom pre obrazec na danom materiáli.

Kamera musí byť orientovaná tak, aby ľavý dolný roh obrazu z kamery zodpovedal pozícii bodu [0,0] laseru.

5.5 Presnosť rezania

Presnosť rezania závisí na viacerých faktoroch:

- Presnosť uloženia ArucoMarkerov na ohraničenie rezacej plochy
- Poloha kamery
- Pohyb krokových motorov
- Presnosť polohy vkladaných obrazcov s ohľadom na vzťah px~mm

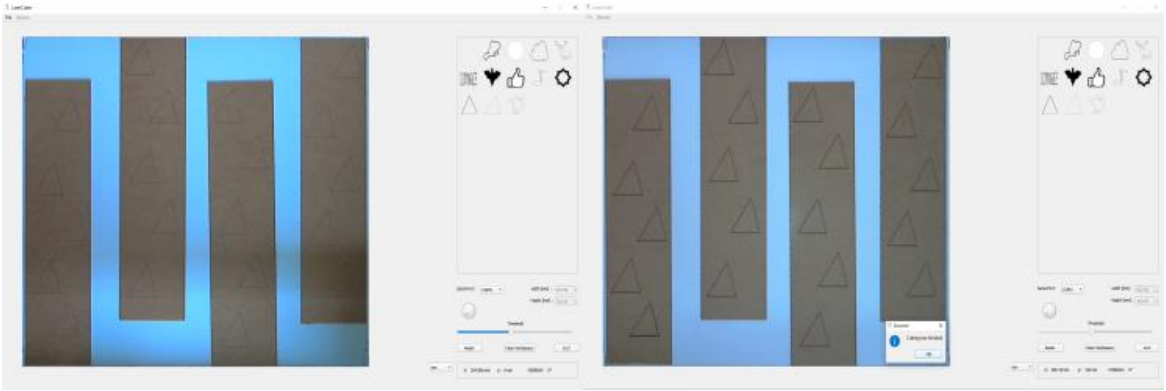
Uloženie ArucoMarkerov by malo byť tak, aby zodpovedalo rozmerom rezacej plochy – rohom. Pri malom pozičnom vychýlení sa zväčší chyba vyrezávaného obrazca.

Poloha kamery má vplyv v zmysle skreslenia a perspektívy kamery. Tým opäť vzniká chyba pozície ArucoMarkerov. Najlepšia pozícia kamery je nad stredom rezacej plochy a výška nad rezacou plochou závisí od FOV parametru kamery.

Pohyb krokových motorov (step~mm) nemusí byť presný v celom rozsahu. Pri väčších vzdialenostiach môže dôjsť k nepresnostiam pohybu.

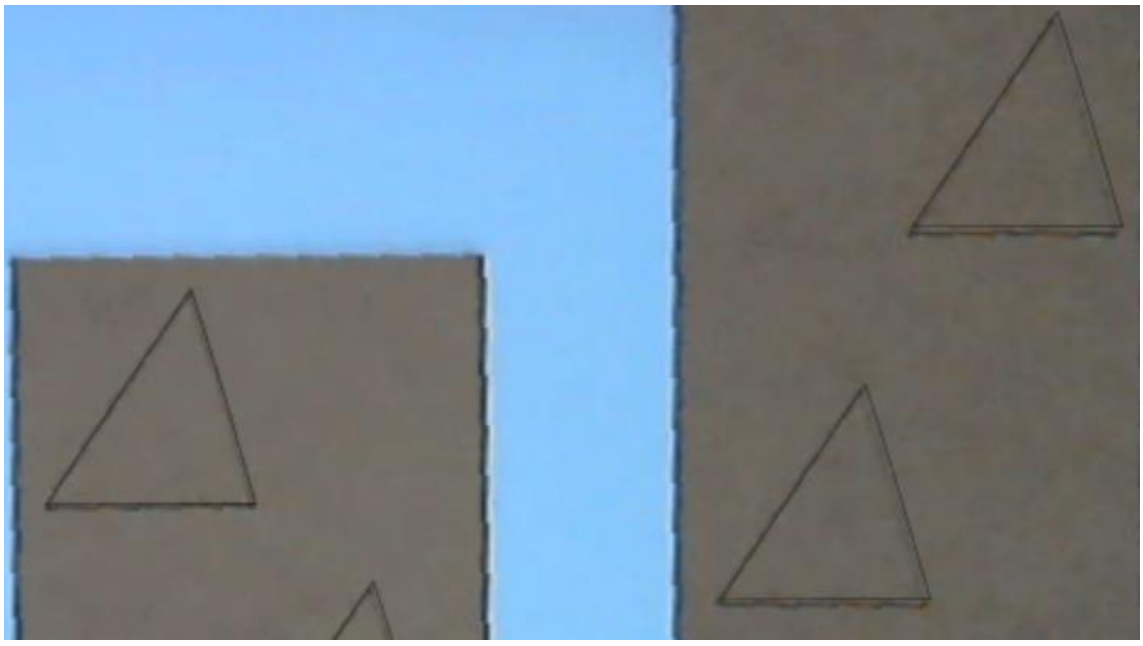
Vzťah px~mm nie je úplne presný a chyba sa môže naakumulovať pri väčších vzdialenostiach.

Pre maximálnu presnosť rezania je teda nevyhnutné, aby ArucoMarkery boli umiestnené čo najpresnejšie v rohoch rezačky. To je možné docieliť pomocou vypálenia obvodu rezacej plochy a následným umiestnením ArucoMarkerov do rohov. ArucoMarkery by mali byť orientované tak, aby konkrétny roh (s indexom 0) bol umiestnený do jednotlivých rohov rezacej plochy.



Obrázok 5-29 - Ukážka presnosti rezania

Pri dodržaní podmienok na čo najväčšiu presnosť sú výsledné obrazce vyrezávané s minimálnou chybou.



Obrázok 5-30 - Detail chyby

Najväčšia chyba vzniká v bodoch najviac vzdialených od počiatku rezacej plochy ([0,0] lasera) – diagonálne od počiatku. Chyba je pravdepodobne spôsobená krokovými motormi rezačky, avšak veľkosť chyby je približne 1mm, čo je zanedbateľné.

6 PRÁCA DO BUDÚCNA

Práca mala pôvodne obsahovať aj možnosť zmeny veľkosti jednotlivých obrazcov. Avšak pri nutnosti modifikácie SVG súborov, v zásade, manuálne sa upustilo od tejto možnosti. Spôsob modifikácie by vtedy pozostával z násobení jednotlivých transformačných matíc. Pri zmene veľkosti by sa zmenil stred otáčania pre daný obrazec, a tak by sa muselo programovo tento stred určovať a korektne rotovať obrazec okolo nového stredu. Za sebou nasledujúce transformácie (matice) by sa násobili medzi sebou. V Qt je tento proces implementačne náročný, nie ani pre maticové násobenia, ale skôr pre spôsob transformácie obrazcov a získavanie správnych parametrov pre tieto matice.

Obohatiť GUI o možnosť väčšej interakcie s rezačkou. Od jednoduchých pohybov v obidvoch osiach, zmeny intenzity lasera až po funkcionality pre laser - zapnutie/vypnutie, „blink“ funkcia a podobne.

Približovanie prenosu kamery by mohlo byť prínosné, ak by bola vyžadovaná väčšia precíznosť pri umiestňovaní objektov. Momentálne to je obmedzené na vzťah px~mm, čo môže byť v rádoch desatinách milimetrov.

Zmiernenie obmedzenia na SVG súbory. Možnosť používať čo najviac rôznych inštancií SVG súborov. V zásade implementovať robustný parser SVG súborov.

Implementácia prvkov grafického editoru, ako napríklad mierka/pravítko po osách zobrazovania, aby mal užívateľ predstavu o rozmeroch a pozícii uloženia obrazcov.

Použiť komplexnejšiu formu trekovania, ktorá by umožnila umiestnenie viacero materiálov na rezaciu plochu a zakotvenie obrazcov k daným materiálom.

Možnosť inverzie logiky programu tak, aby pozadie, podklad, rezačky mohol byť tmavého odtieňa a rezalo by sa na materiál svetlejšej farby ako pozadie. Zvýšila by sa tak robustnosť riešenia.

Iný prospekt by bola implementácia optimalizácie cesty rezania, tak aby bol čas rezania minimalizovaný. Rovnako aj komunikovať spätnú väzbu o priebehu procesu rezania.

7 ZÁVER

Zadaním bolo implementovať grafické užívateľské rozhranie pre laserovú rezačku s prvkami rozšírenej reality. Jedná sa o komerčnejší prístup k problematike s dôrazom na bezpečnosť a pohodlnosť používateľa. K realizácii bolo potrebné vykonať rešerš v oblasti rozšírenej reality, konkrétnejšie v oblasti užívateľských rozhraní. Rovnako bolo nutné porozumieť problematike počítačového videnia a programovania v Qt, v ktorom sa rozhranie vyvíjalo. V prvom rade sa musel implementovať grafický editor pre SVG súbory, kde došlo k niekoľkým prekážkam a muselo sa upustiť od niektorých prvkov. Následne sa využili nástroje počítačového videnia, knižnice OpenCV, na naplnenie prvkov rozšírenej reality. Podstatná bola taktiež konfigurácia, použitie firmvéru GRBL a konverzia SVG na G-Code pre realizáciu rezania.

Boli navrhnuté dva spôsoby riešenia pre detekciu rezačky a podstatných parametrov na realizáciu. Prvý spôsob by využíval kalibráciu kamery, a umiestnenie detekčných značiek na rohy rámu rezačky, a druhý, ktorý bol použitý, umiestni ArucoMarkery na rohy rezacej plochy. Pri tomto spôsobe bolo nevyhnutné vyrobiť pomôcky na uloženie týchto značiek na správne miesto, tak, aby boli stále pevne spojené s rezačkou.

Pri správnom nastavení a pozičnom uložení všetkých prvkov sa riešenie ukázalo ako veľmi presné. Je však potrebné vykonať súbor inicializačných krokov, jednoduchých na realizáciu aj pre človeka s netechnickým pozadím, pre prípravu scény na rezanie. Výsledky rezania sú prezentované v kapitole 5.5.

Použité prístroje

Laboratórny zdroj Basetech BT-305; WEEE-Reg-Nr. DE28001718 ; Serial No. 0677
Webkamera Logitech C270
Riadiaca doska Arduino Nano, mikrokontrolér ATmega328P
Krokové motory Nema 17
USB kábel 1x
Mini-USB kábel 1x

Referencie

[1] R. H. T. M. Kristoffer Winge, „VAL: Visually Augmented Laser cutting to enhance and support creativity,“ rev. *2014 IEEE International Symposium on Mixed and Augmented Reality - Media, Art, Social Science, Humanities and Design (ISMAR-MASH'D)*, Munich, Germany , 2014.

[2] J. G. C. L. Alex Olwal, „Spatial Augmented Reality on Industrial CNC Machines,“ rev. *The International Society for Optical Engineering 6804*, 2008.

[3] T.-H. K. Yunbo Zhang, „sciencedirect.com,“ 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978918308163. [Cit. 21 October 2020].

[4] A. C. a. G. L. Mark Billinghurst, „Survey of Augmented Reality,“ *Foundations and Trends in Human-Computer Interaction.*, zv. 8, %1. vyd.2-3, pp. 73-272, 2008.

[5] S. S. Yahya Ghazwani, „Interaction in Augmented Reality: Challenges to Enhance User Experience,“ rev. *ICVARS 2020: Proceedings of the 2020 4th International Conference on Virtual and Augmented Reality Simulations*, Sydney, AUS, 2020.

[6] J. Rekimoto, „Navicam: A magnifying glass approach to augmented reality,“ *Presence: Teleoperators and Virtual Environments*, August 1997.

[7] R. G. Mark Billinghurst, „researchgate.net,“ February 2005. [Online]. Available: https://www.researchgate.net/publication/234795383_Designing_augmented_reality_interfaces. [Cit. 28 October 2020].

[8] H. B.-L. D. M. B. Feng Zhou, „Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR,“ rev. *IEEE International Symposium on Mixed and Augmented Reality 2008*, Cambridge, UK, 2008.

[9] M. B. G. B. H. B.-L. D. G. F. W. Kangsoo Kim, „Revisiting Trends in Augmented Reality Research: A Review of the 2nd Decade of ISMAR (2008–2017),“ IEEE, 2018.

[10] V. H. R. B. Milan Sonka, Image processing, Analysis, and Machine Vision, Cengage learning, 2008.

[11] K. Plataniotis, „researchgate.net,“ 2006. [Online]. Available: https://www.researchgate.net/publication/228638268_Color_image_denoising_using_evolutionary_computation. [Cit. 26 December 2020].

J. K. M. K. S. FuS, „sciencedirect.com,“ 1981. [Online]. Available: [12] https://www.sciencedirect.com/science/article/abs/pii/0031320381900285. [Cit. 5 November 2020].

A. Z. Richard Hartley, Multiple View Geometry in Computer Vision, [13] Cambridge: Cambridge University Press , 2004.

H. Z. N. H. J. K. Xiangjian He, „Estimation of Internal and External [14] Parameters for Camera Calibration Using 1D Pattern,“ rev. *Advanced Video and Signal Based Surveillance, 2006 IEEE International* , Sydney, Australia, 2006.

„mathworks.com,“ [Online]. Available: [15] https://www.mathworks.com/help/vision/ug/camera-calibration.html. [Cit. 26 12 2020].

l. University of Toronto, *Camera Models and Parameters,* Toronto: [16] University of Toronto.

B. F. M. A. P. C. E. D. &. M. I. Julie Carmigniani, „Augmented reality [17] technologies, systems and applications,“ 14 December 2010. [Online]. Available: https://link.springer.com/article/10.1007/s11042-010-0660-6. [Cit. 28 October 2020].

N. R. P. S. K. PAL, „sciencedirect.com,“ September 1993. [Online]. [18] Available: https://www.sciencedirect.com/science/article/abs/pii/003132039390135J. [Cit. 5 November 2020].

G. Wolberg, „Geometric Transformation Techniques for Digital Images: A [19] Survey,“ Department of Computer Science, Columbia University, New York, 1988.

R. Szeliski, Computer Vision: Algorithms and Applications, Springer, [20] 2010.

M. F. M. M. M. Sharifi, „A classified and comparative study of edge [21] detection algorithms,“ rev. *Proceedings. International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, USA, 2002.

„https://www.di.univr.it/,“ [Online]. Available: [22] https://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid125113.pdf. [Cit. 4 November 2020].

Zoznam príloh

Prílohou práce je SD karta s elektronickou verziou práce vo formáte pdf. Na SD karte sa nachádza :

- Zdrojové kódy + dokumentácia
- Výsledná zložka aplikácie
- Tabuľka konfigurácie GRBL
- Vygenerované ArucoMarkery použité na realizáciu